大数据管理实验代码

大数据2101班 李嘉鹏 U202115652

Lab 1

1.

```
select *
from business
where business_info->'$.city'="Tampa"
order by business_info->'$.review_count' desc
limit 10;
```

2.

```
select
json_keys(business_info) as keys_info,
json_length(business_info) as key_num_info,
json_keys(business_info, '$.attributes') as keys_attr,
json_length(business_info, '$.attributes') as key_num_attr
from business
limit 5;
```

3.

```
select
business_info->'$.name' as name,
json_type(business_info->'$.name') as name_type,
business_info->'$.stars' as stars,
json_type(business_info->'$.stars') as stars_type,
business_info->'$.attributes' as attributes,
json_type(business_info->'$. attributes') as attributes_type
from business
limit 5;
```

4.

```
select
```

```
json_unquote(business_info->'$.name') as name,
business_info->'$.attributes' as attributes,
business_info->'$.hours' as open_time
from business
where business_info->'$.attributes.HasTV'="True"
and (business_info->'$.hours' is null or business_info->'$.hours.Sunday' is null)
order by name
limit 10;
```

5.

```
explain format=json
select * from user where user_info->'$.name'='Wanda';
```

使用MongoDB查询同样的语句，并使用explain查看查询计划：

```
db.user.find({'name':"Wanda"}).explain("executionStats")
```

6.

```
select json_pretty(business_info)
from business
where business_id="4r3Ck65DCG1T6gpWodPyrg";
```

更新操作：

```
update business
set business_info=json_set(business_info, '$.hours.Tuesday', "16:0-23:0", '$.stars', 4.5, '$.WiFi', "Free")
where business_id="4r3Ck65DCG1T6gpWodPyrg";
```

再次查询：

```
select json_pretty(business_info)
from business
where business_id="4r3Ck65DCG1T6gpWodPyrg";
```

7.

```
insert into business(business_id, business_info)
select "aaaaaabbbbbbcccccc2023", business_info from business where business_id='5d-fkQteaqO6CSCqS5q4rw';

update business
set business_info=json_remove(business_info, '$.name')
where business_id="aaaaaabbbbbbcccccc2023";
```

查询:

```
select json_pretty(business_info)
from business
where business_id="aaaaaabbbbbbcccccc2023";
```

8.

```
select
state,
json_objectagg(city, count) as city_occ_num
from
(
select
business_info->'$.state' as state,
json_unquote(business_info->'$.city') as city,
count(*) as count
from business
group by state, city
) as sub
group by state
order by state;
```

9.

```
select sub.userid as user_id,
uu.user_info->'$.name' as name,
```

```
sub.textarray as text_array from
(
select t.user_id as userid,
json_arrayagg(t.tip_info->'$.text') as textarray
from tip t
join user u on REGEXP_LIKE(u.user_info->'$.friends', t.user_id)
where u.user_id='__1cb6cwl3uAbMTK3xaGbg'
group by t.user_id
) as sub, user uu
where uu.user_id = sub.userid
order by name;
```

10.

```
select
a.business_info->'$.name' as name1,
a.business_info->'$.city' as city1,
b.business_info->'$.name' as name2,
b.business_info->'$.city' as city2,
a.business_info->'$.hours' as hours1,
b.business_info->'$.hours' as hours2,
JSON_OVERLAPS(a.business_info->'$.hours', b.business_info->'$.hours') as has_same_opentime
from business a
join business b
on a.business_info->'$.city' = 'EdMonton' and b.business_info->'$.city' = 'Elsmere';
```

11.

```
select
user_info->'$.name' as name,
user_info->'$.average_stars' as avg_stars,
JSON_ARRAY(user_info->'$.funny', user_info->'$.useful', user_info->'$.cool', user_info->'$.funny'+user_info->'$.useful'+user_info->'$.cool') as '[funny,useful,cool,sum]'
from user
where user_info->'$.funny'>2000
and user_info->'$.average_stars'>4.0
```

```
order by user_info->'$.average_stars' desc
limit 10;
```

报错：

手动修改缓存池大小即可

在前面加上explain format=json

和第一题对比：

12.

```
select
json_merge_preserve(b.business_info, u.user_info)
from
(
select business_id,
count(*) as count
from tip
group by business_id
order by count desc
limit 1
) as tb
join
(
select user_id,
count(*) as count
from tip
group by user_id
order by count desc
limit 1
) as tu
join business b on tb.business_id=b.business_id
join user u on tu.user_id=u.user_id;
```

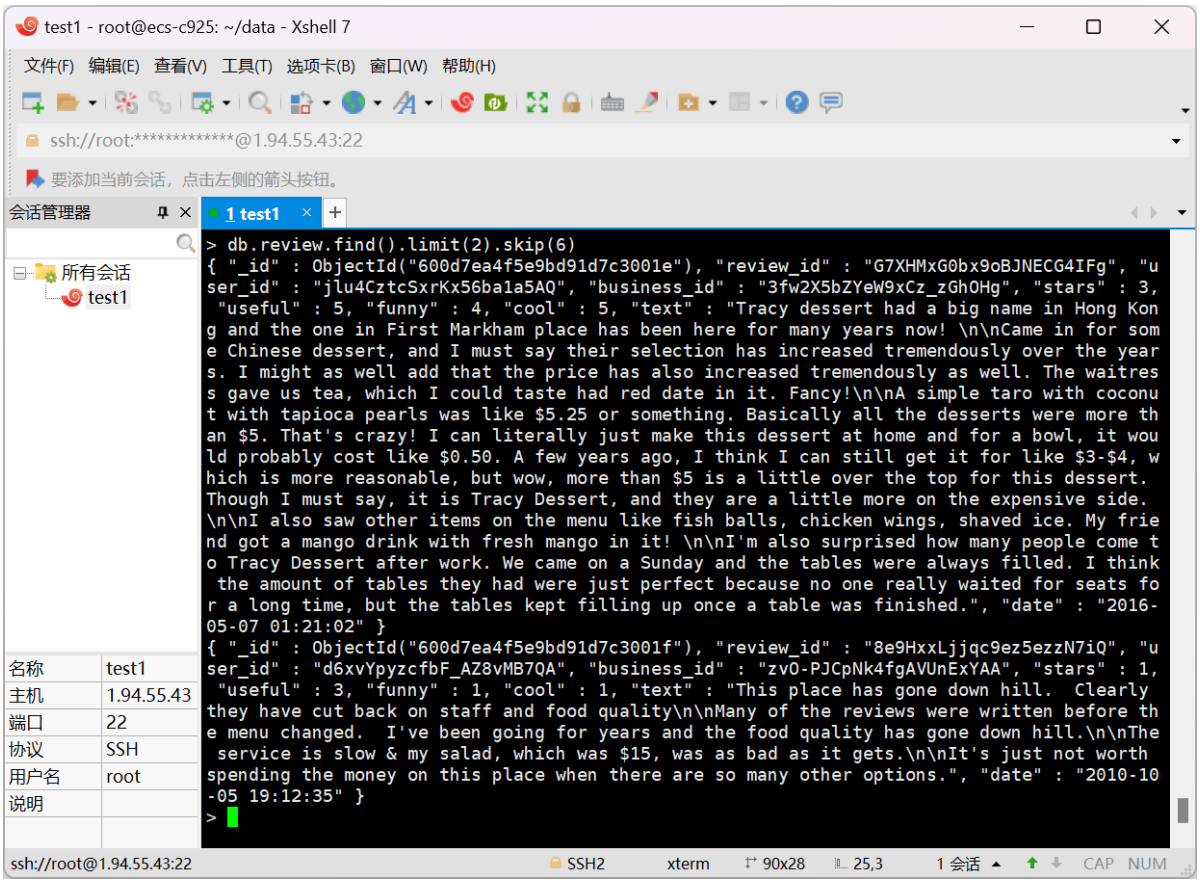
13.

```
select
  sub.business_name,
  sub.review_count as business_review_count,
case when sub.hours->'$.Tuesday' is null then 0 else 1 end as business_open_on_Tuesday,
  jt.time_slot
from
  (select
     business_info->'$.name' as business_name,
     business_info->'$.review_count' as review_count,
     business_info->'$.hours' as hours
   from business
   order by business_info->'$.review_count' desc
   limit 3) as sub
join json_table(
    sub.hours,
    "$.*" columns (
      time_slot VARCHAR(255) PATH '$')
  ) as jt on 1=1
order by sub.business_name;
```

Lab 2

1.

```
db.review.find().limit(2).skip(6)
```



2.

db.business.find({'city':'Las Vegas'}).limit(5)

3.

db.user.find({'name':'Steve'},{'useful':1,'cool':1}).limit(10)

```
> db.user.find({'name':'Steve'},{'useful':1,'cool':1}).limit(10)
{ "_id" : ObjectId("600d35c72cb85d6290190f0c"), "useful" : 680, "cool" : 383 }
{ "_id" : ObjectId("600d35c72cb85d62901911d1"), "useful" : 33, "cool" : 10 }
{ "_id" : ObjectId("600d35c72cb85d629019141f"), "useful" : 4359, "cool" : 3663 }
{ "_id" : ObjectId("600d35c72cb85d62901914b4"), "useful" : 4, "cool" : 2 }
{ "_id" : ObjectId("600d35c72cb85d6290191803"), "useful" : 31343, "cool" : 28316 }
{ "_id" : ObjectId("600d35c72cb85d6290191a86"), "useful" : 2, "cool" : 2 }
{ "_id" : ObjectId("600d35c72cb85d6290191ad0"), "useful" : 9, "cool" : 0 }
{ "_id" : ObjectId("600d35c72cb85d6290191b18"), "useful" : 709, "cool" : 339 }
{ "_id" : ObjectId("600d35c72cb85d6290191b88"), "useful" : 295, "cool" : 115 }
{ "_id" : ObjectId("600d35c72cb85d6290191bb0"), "useful" : 41, "cool" : 8 }
```

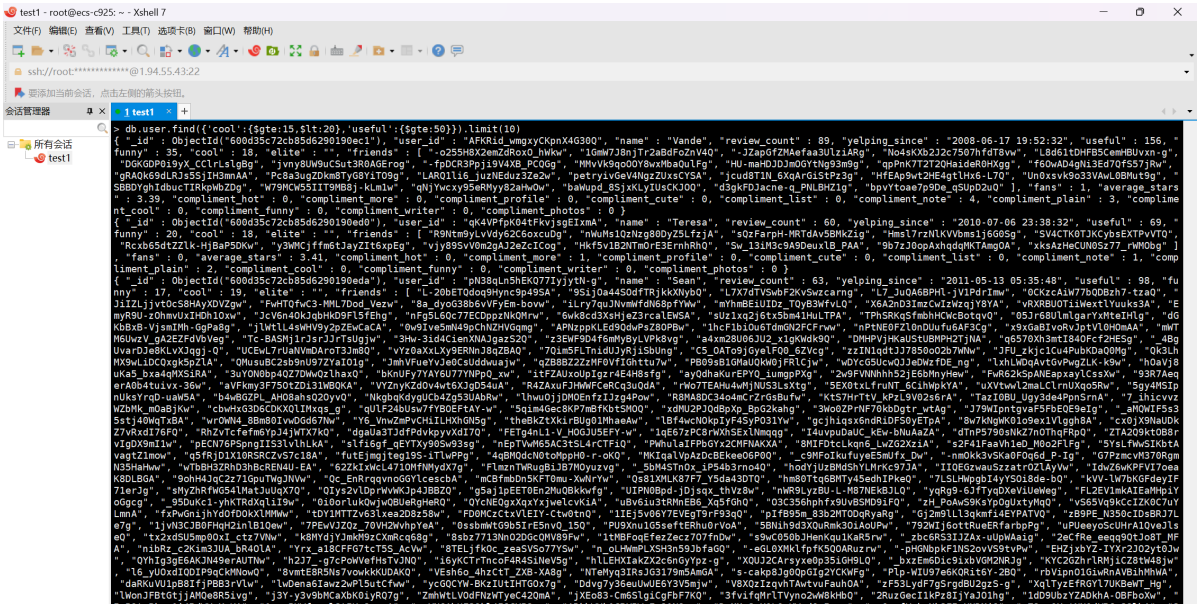
4.

db.user.find({'funny':{$in:[66,67,68]}},{'name':1,'funny':1}).limit(20)

```
> db.user.find({'funny':{$in:[66,67,68]}},{'name':1,'funny':1}).limit(20)
{ "_id" : ObjectId("600d35c72cb85d62901910ac"), "name" : "Aileen", "funny" : 68 }
{ "_id" : ObjectId("600d35c72cb85d62901911bc"), "name" : "Alison", "funny" : 67 }
{ "_id" : ObjectId("600d35c72cb85d62901911b6"), "name" : "Karin", "funny" : 66 }
{ "_id" : ObjectId("600d35c72cb85d6290191290"), "name" : "Ashlee", "funny" : 68 }
{ "_id" : ObjectId("600d35c72cb85d62901913ea"), "name" : "Angie", "funny" : 66 }
{ "_id" : ObjectId("600d35c72cb85d62901914b6"), "name" : "Kristen", "funny" : 66 }
{ "_id" : ObjectId("600d35c72cb85d6290191840"), "name" : "Kathleen", "funny" : 67 }
{ "_id" : ObjectId("600d35c72cb85d62901918f3"), "name" : "Ted", "funny" : 67 }
{ "_id" : ObjectId("600d35c72cb85d6290191907"), "name" : "Donald", "funny" : 66 }
{ "_id" : ObjectId("600d35c72cb85d6290191911"), "name" : "Heidi", "funny" : 67 }
{ "_id" : ObjectId("600d35c72cb85d6290191953"), "name" : "Angelica", "funny" : 66 }
{ "_id" : ObjectId("600d35c72cb85d62901919da"), "name" : "Julia", "funny" : 68 }
{ "_id" : ObjectId("600d35c72cb85d6290191a2e"), "name" : "Melinda", "funny" : 68 }
{ "_id" : ObjectId("600d35c72cb85d6290191b03"), "name" : "Lisa", "funny" : 68 }
{ "_id" : ObjectId("600d35c72cb85d6290191b45"), "name" : "Sofia", "funny" : 68 }
{ "_id" : ObjectId("600d35c72cb85d6290191bbd"), "name" : "Amy", "funny" : 68 }
{ "_id" : ObjectId("600d35c72cb85d6290191c77"), "name" : "Keane", "funny" : 67 }
{ "_id" : ObjectId("600d35c72cb85d6290191ce2"), "name" : "Karina", "funny" : 66 }
{ "_id" : ObjectId("600d35c72cb85d6290191f29"), "name" : "Jess", "funny" : 66 }
{ "_id" : ObjectId("600d35c72cb85d629019203d"), "name" : "Linda", "funny" : 68 }
```

5.

db.user.find({'cool':{$gte:15,$lt:20},'useful':{$gte:50}}).limit(10)



6.

db.business.aggregate({$group:{_id:'',total:{$sum:1}}})

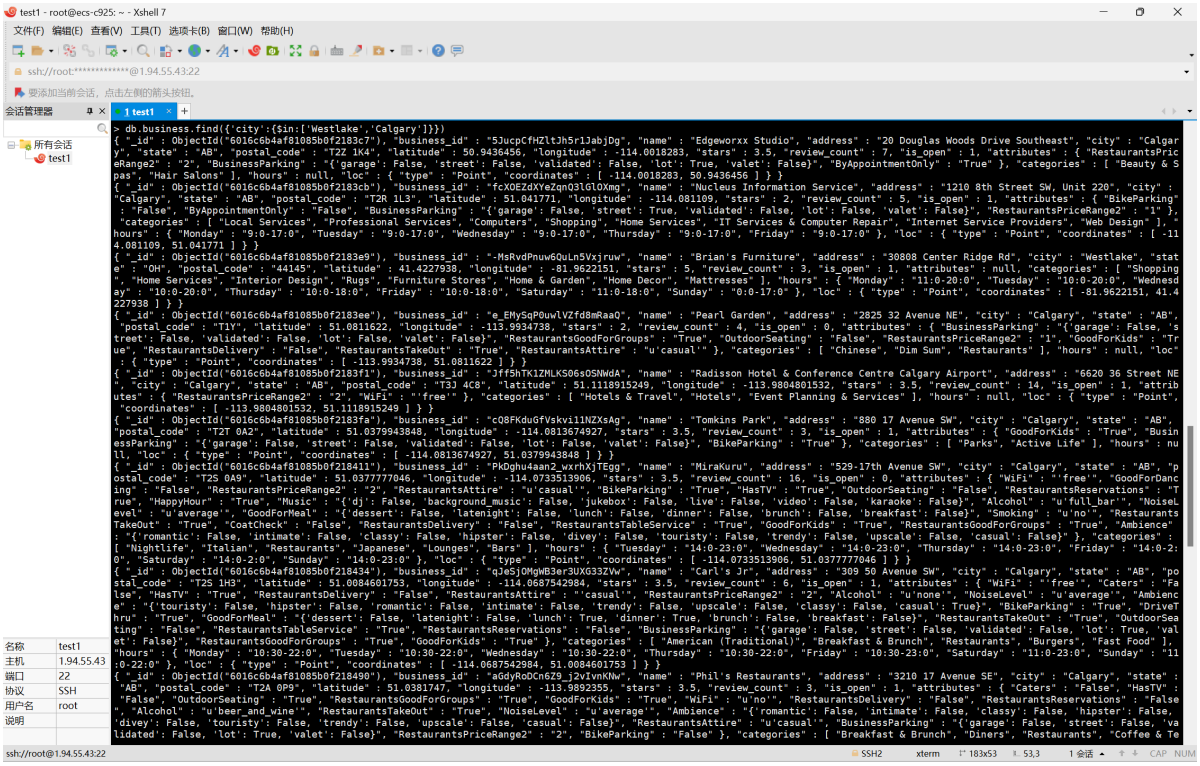
```
> db.business.aggregate({$group:{_id:'',total:{$sum:1}}})
{ "_id" : "", "total" : 192609 }
```

db.business.explain("executionStats").count()

```
> db.business.explain("executionStats").count()
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "yelp.business",
                "indexFilterSet" : false,
                "winningPlan" : {
                        "stage" : "RECORD_STORE_FAST_COUNT"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 0,
                "executionTimeMillis" : 0,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 0,
                "executionStages" : {
                        "stage" : "RECORD_STORE_FAST_COUNT",
                        "nReturned" : 0,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 1,
                        "advanced" : 0,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "nCounted" : 192609,
                        "nSkipped" : 0
                }
        },
        "serverInfo" : {
                "host" : "ecs-c925",
                "port" : 27017,
                "version" : "4.4.17",
                "gitVersion" : "85de0cc83f4dc64dbbac7fe028a4866228c1b5d1"
        },
        "ok" : 1
}
```

7.

db.business.find({'city':{$in:['Westlake','Calgary']}})



```
> db.business.find({'city':{$in:['Westlake','Calgary']}}).count()
8125
```

8.

db.business.find({'categories':{$size:6}},{'_id':1,'categories':1}).limit(10)

```
> db.business.find({'categories':{$size:6}},{'_id':1,'categories':1}).limit(10)
{ "_id" : ObjectId("6016c6b4af81085b0f2183cc"), "categories" : [ "Restaurants", "Breakfast & Brunch", "Mexican", "Tacos", "Tex-Mex", "Fast Food" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f2183cd"), "categories" : [ "Bars", "Nightlife", "Pubs", "Barbers", "Beauty & Spas", "Irish Pub" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f2183d4"), "categories" : [ "Trainers", "Health & Medical", "Active Life", "Physical Therapy", "Gyms", "Fitness & Instruction" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f2183d7"), "categories" : [ "Nightlife", "Arts & Entertainment", "Bars", "Strip Clubs", "Adult Entertainment", "Dance Clubs" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f2183da"), "categories" : [ "Mexican", "Restaurants", "Patisserie/Cake Shop", "Food", "Bars", "Nightlife" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f2183e8"), "categories" : [ "Auto Glass Services", "Auto Detailing", "Automotive", "Auto Parts & Supplies", "Auto Customization", "Vehicle Wraps"
] }
{ "_id" : ObjectId("6016c6b4af81085b0f2183f3"), "categories" : [ "Restaurants", "Breakfast & Brunch", "Bars", "Modern European", "Nightlife", "Wine Bars" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f218400"), "categories" : [ "Men's Clothing", "Sporting Goods", "Shopping", "Fashion", "Women's Clothing", "Sports Wear" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f218403"), "categories" : [ "Nightlife", "Bars", "Polish", "Modern European", "Restaurants", "Vegan" ] }
{ "_id" : ObjectId("6016c6b4af81085b0f218411"), "categories" : [ "Nightlife", "Italian", "Restaurants", "Japanese", "Lounges", "Bars" ] }
```

9.

db.business.find({ business_id: "5JucpCfHZltJh5r1JabjDg" }).explain("executionStats")

```
> db.business.find({ business_id: "5JucpCfHZltJh5r1JabjDg" }).explain("executionStats");
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "yelp.business",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "business_id" : {
                                "$eq" : "5JucpCfHZltJh5r1JabjDg"
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "business_id" : {
                                        "$eq" : "5JucpCfHZltJh5r1JabjDg"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 1,
                "executionTimeMillis" : 64,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 192609,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "business_id" : {
                                        "$eq" : "5JucpCfHZltJh5r1JabjDg"
                                }
                        },
                        "nReturned" : 1,
                        "executionTimeMillisEstimate" : 6,
                        "works" : 192611,
                        "advanced" : 1,
                        "needTime" : 192609,
                        "needYield" : 0,
                        "saveState" : 192,
                        "restoreState" : 192,
                        "isEOF" : 1,
                        "direction" : "forward",
                        "docsExamined" : 192609
                }
        },
        "serverInfo" : {
                "host" : "ecs-c925",
                "port" : 27017,
                "version" : "4.4.17",
                "gitVersion" : "85de0cc83f4dc64dbbac7fe028a4866228c1b5d1"
```

可以看出使用了COLLSCAN方式（遍历整个集合）

执行计划关注parsedQuery（解析的查询表达式）、winningPlan（查询执行策略）、rejectedPlans（考虑但未选中的执行策略）

通过物理方式进行优化：为business_id建立索引。

db.business.createIndex({'business_id':1})

```
> db.business.createIndex({'business_id':1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

再次使用explain查看执行过程：

```
> db.business.find({ business_id: "5JucpCfHZltJh5r1JabjDg" }).explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "yelp.business",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "business_id" : {
                                "$eq" : "5JucpCfHZltJh5r1JabjDg"
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "business_id" : 1
                                },
                                "indexName" : "business_id_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "business_id" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "business_id" : [
                                                "[\"5JucpCfHZltJh5r1JabjDg\", \"5JucpCfHZltJh5r1JabjDg\"]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 1,
                "executionTimeMillis" : 0,
                "totalKeysExamined" : 1,
                "totalDocsExamined" : 1,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 1,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 2,
                        "advanced" : 1,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
```

```
                        "isEOF" : 1,
                        "docsExamined" : 1,
                        "alreadyHasObj" : 0,
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "nReturned" : 1,
                                "executionTimeMillisEstimate" : 0,
                                "works" : 2,
                                "advanced" : 1,
                                "needTime" : 0,
                                "needYield" : 0,
                                "saveState" : 0,
                                "restoreState" : 0,
                                "isEOF" : 1,
                                "keyPattern" : {
                                        "business_id" : 1
                                },
                                "indexName" : "business_id_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "business_id" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "business_id" : [
                                                "[\"5JucpCfHZltJh5r1JabjDg\", \"5JucpCfHZltJh5r1JabjDg\"]"
                                        ]
                                },
                                "keysExamined" : 1,
                                "seeks" : 1,
                                "dupsTested" : 0,
                                "dupsDropped" : 0
                        }
                }
        },
        "serverInfo" : {
                "host" : "ecs-c925",
                "port" : 27017,
                "version" : "4.4.17",
                "gitVersion" : "85de0cc83f4dc64dbbac7fe028a4866228c1b5d1"
        },
        "ok" : 1
}
```

很明显可以看到时间变短（64ms->0ms），且查询方式变为FETCH，索引名为business_id_1

10.

db.business.aggregate([

{$group:{cnt:{$sum:1}, '_id':'$stars'}},

{$sort:{'_id': -1}}])

```
> db.business.aggregate([ {$group:{cnt:{$sum:1}, '_id':'$stars'}}, {$sort:{'_id': -1}}])
{ "_id" : 5, "cnt" : 28216 }
{ "_id" : 4.5, "cnt" : 27301 }
{ "_id" : 4, "cnt" : 35969 }
{ "_id" : 3.5, "cnt" : 35008 }
{ "_id" : 3, "cnt" : 25996 }
{ "_id" : 2.5, "cnt" : 18843 }
{ "_id" : 2, "cnt" : 11426 }
{ "_id" : 1.5, "cnt" : 4976 }
{ "_id" : 1, "cnt" : 4874 }
```

11.

db.review.aggregate([{$limit: 500000}, {$out: "Subreview"}])

```
> db.review.aggregate([{$limit: 500000}, {$out: "Subreview"}])
> show collections
Subreview
business
review
test_map_reduce
user
>
> db.Subreview.count()
500000
```

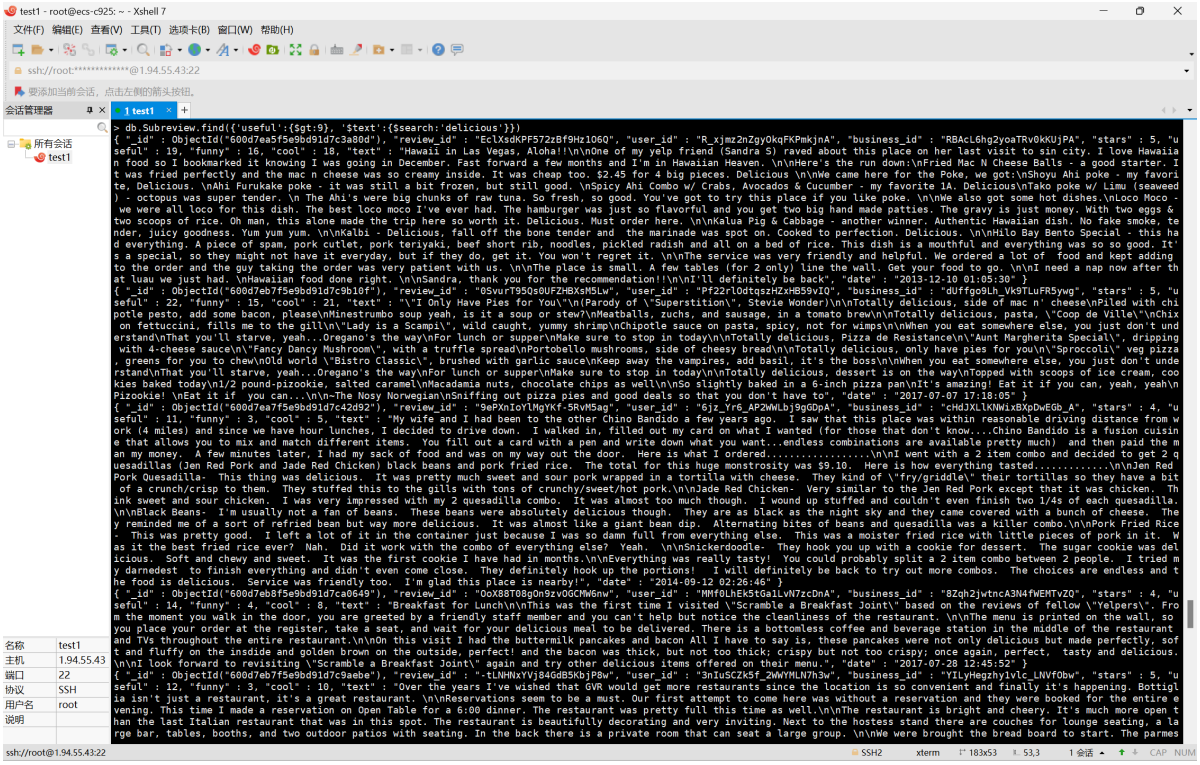
db.Subreview.createIndex({'text':'text'})

```
> db.Subreview.createIndex({'text':'text'})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

db.Subreview.createIndex({'useful':1})

```
> db.Subreview.createIndex({'useful':1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

db.Subreview.find({'useful':{$gt:9}, '$text':{$search:'delicious'}})




```
> db.Subreview.find({'useful':{$gt:9}, '$text':{$search:'delicious'}}).count()
911
```

12.

db.Subreview.aggregate([
{$match:{'useful':{$gt:6}, 'funny':{$gt:6}, 'cool':{$gt:6}}},
{$group:{*id:'$business_id', stars_avg:{$avg:'$stars'}}},*
*{$sort:{*id:-1}}
])

```
> db.Subreview.aggregate([ {$match:{'useful':{$gt:6}, 'funny':{$gt:6}, 'cool':{$gt:6}}}, {
$gdb.Subreview.aggregate([ {$match:{'useful':{$gt:6}, 'funny':{$gt:6}, 'cool':{$gt:6}}}, {
$group:{_id:'$business_id', stars_avg:{$avg:'$stars'}}}, {$sort:{_id:-1}} ])
{ "_id" : "zvQIEpJUmLLmMMffNntHXQ", "stars_avg" : 2 }
{ "_id" : "zrYpLdnGKA_EmOhgRCy_vg", "stars_avg" : 4.5 }
{ "_id" : "znWHLW1pt19HzW1VY6KfCA", "stars_avg" : 4 }
{ "_id" : "znBAFVFRMvdUvJ7eJY_vjA", "stars_avg" : 3 }
{ "_id" : "zitXLajbETOuHQJfok4a4g", "stars_avg" : 4 }
{ "_id" : "zidkKI_N1OPxsiddTOQH_Q", "stars_avg" : 4 }
{ "_id" : "zenxm7shqn6OCfw_2PGbrg", "stars_avg" : 4 }
{ "_id" : "zcpfOE1tBET_5YUjq8y11g", "stars_avg" : 4 }
{ "_id" : "zauhMY78k36XPfxD3GURkQ", "stars_avg" : 3.5 }
{ "_id" : "zYRBRfYuq_6x-wNKa8NqrA", "stars_avg" : 4 }
{ "_id" : "zYKblyjRDoiOum6NYuvZ6g", "stars_avg" : 1 }
{ "_id" : "zWB6hcE7Pxww0VoI2fYIWw", "stars_avg" : 4 }
{ "_id" : "zSc_PmocVDJEtQwin0ts2w", "stars_avg" : 3 }
{ "_id" : "zPxWTy_2WOslWtKCr8B1nQ", "stars_avg" : 1 }
{ "_id" : "zKqWjDZHo-lk7_E3hHkORg", "stars_avg" : 4 }
{ "_id" : "zHJhV-P_FesPreukKMo0eA", "stars_avg" : 1.5 }
{ "_id" : "zEq6RxjGkyf9AzUFtBvcOg", "stars_avg" : 4 }
{ "_id" : "zBtR7328Vuts_7B9qm3DVQ", "stars_avg" : 5 }
{ "_id" : "z9oJeVmNEc3F0ToZ0x4WuQ", "stars_avg" : 4.666666666666667 }
{ "_id" : "z8Em-bhZI3Mmspml7tj6tg", "stars_avg" : 2 }
Type "it" for more
```

统计结果数量：

db.Subreview.aggregate([
{$match:{'useful':{$gt:6}, 'funny':{$gt:6}, 'cool':{$gt:6}}},
{$group:{*id:'$business_id', stars_avg:{$avg:'$stars'}}},*
*{$sort:{*id:-1}},

{$count: "TotalCount"}

])

```
> db.Subreview.aggregate([ {$match:{'useful':{$gt:6}, 'funny':{$gt:6}, 'cool':{$gt:6}}}, {
$group:{_id:'$business_id', stars_avg:{$avg:'$stars'}}}, {$sort:{_id:-1}}, {$count: "Total
Count"} ])
{ "TotalCount" : 1558 }
```

13.

db.business.createIndex({'loc':"2dsphere"})

```
> db.business.createIndex({'loc':"2dsphere"})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

查询目标商家的坐标：

```
> db.business.find({'business_id':'xvX2CttrVhyG2z1dFg_0xw'})
{ "_id" : ObjectId("6016c6b4af81085b0f2183c4"), "business_id" : "xvX2CttrVhyG2z1dFg_0xw",
"name" : "Farmers Insurance - Paul Lorenz", "address" : "15655 W Roosevelt St, Ste 237", "
city" : "Goodyear", "state" : "AZ", "postal_code" : "85338", "latitude" : 33.4556129678, "
longitude" : -112.3955963552, "stars" : 5, "review_count" : 3, "is_open" : 1, "attributes"
 : null, "categories" : [ "Insurance", "Financial Services" ], "hours" : { "Monday" : "8:0
-17:0", "Tuesday" : "8:0-17:0", "Wednesday" : "8:0-17:0", "Thursday" : "8:0-17:0", "Friday
" : "8:0-17:0" }, "loc" : { "type" : "Point", "coordinates" : [ -112.3955963552, 33.455612
9678 ] } }
```

db.business.find(

{loc: {$near:{$geometry:{type:"Point", coordinates:[-112.3955963552, 33.4556129678]},

$maxDistance: 100}}},

{'name':1, 'address':1, 'stars':1, '_id':0}

)

```
> db.business.find( {loc: {$near:{$geometry:{type:"Point", coordinates:[-112.3955963552, 3
3.4556129678]}, $maxDistance: 100}}}, {'name':1, 'address':1, 'stars':1, '_id':0} )
{ "name" : "Farmers Insurance - Paul Lorenz", "address" : "15655 W Roosevelt St, Ste 237",
 "stars" : 5 }
{ "name" : "Kurt Bojarski -  American Family Insurance", "address" : "15655 W Roosevelt St
, Ste 101", "stars" : 5 }
```

14.

首先创建一个关于用户id和评论日期的索引

db.Subreview.createIndex({user_id: 1, date: 1})

然后聚合查询

db.Subreview.aggregate([

{$match:{'date':{$gte: "2017-01-01"}}},

{$group:{_id:"$user_id", total:{$sum:1}}},

{$sort: {total: -1}},

{$limit: 20}

])

```
> db.Subreview.aggregate([ {$match:{'date':{$gte: "2017-01-01"}}}, {$group:{_id:"$user_id"
, total:{$sum:1}}}, {$sort: {total: -1}}, {$limit: 20} ])
{ "_id" : "I-4KVZ9lqHhk8469X9FvhA", "total" : 61 }
{ "_id" : "qKpkRCPk4ycbllTfFcRbNw", "total" : 46 }
{ "_id" : "bLbSNkLggFnqwNNzzq-Ijw", "total" : 42 }
{ "_id" : "ic-tyi1jElL_umxZVh8KNA", "total" : 37 }
{ "_id" : "tFSnxUGaNa2pwb4zClBu_Q", "total" : 32 }
{ "_id" : "z9w399cBpCAKXhH_JA1AtQ", "total" : 32 }
{ "_id" : "nyl_1VcRIAyI55bb_scpdw", "total" : 31 }
{ "_id" : "zPByj_3y6TBok5BpCyYrTw", "total" : 31 }
{ "_id" : "YE54kKTuqJJPNYWIKIpOEQ", "total" : 30 }
{ "_id" : "xDl9ZF3SckkZde_48W6WeA", "total" : 29 }
{ "_id" : "PcvbBOCOcs6_suRDH7TSTg", "total" : 29 }
{ "_id" : "f_5VRh79aew1cVWUmC1PJA", "total" : 29 }
{ "_id" : "6Ki3bAL0wx9ymbdJqbSWMA", "total" : 28 }
{ "_id" : "PGeiszoVusiv0wTHVdWklA", "total" : 28 }
{ "_id" : "keBv05MsMFBd0Hu98vXThQ", "total" : 27 }
{ "_id" : "QbDJB-4XAxn3BGwi6Rrzjw", "total" : 26 }
{ "_id" : "wNLZnNNLV8rOGiPjqMPVdQ", "total" : 26 }
{ "_id" : "3nIuSCZk5f_2WWYMLN7h3w", "total" : 25 }
{ "_id" : "ELcQDlf69kb-ihJfxZyL0A", "total" : 25 }
{ "_id" : "U95wccXN_J8JwA5Ktlu8tw", "total" : 25 }
```

15.

db.business.mapReduce(

```
function(){
emit(this.business_id, {stars: this.stars, count:1});
},

function(key, values){
var totalStars=0;
var totalCount=0;
values.forEach(function(value) {
totalStars+=value.stars;
totalCount+=value.count;
});
return {stars:totalStars, count:totalCount};
},
{
out:"test_map_reduce",
finalize: function(key, reducedValue){
var avgStars = reducedValue.stars / reducedValue.count;
return { count: reducedValue.count, stars: reducedValue.stars, avg: avgStars };
}
}
)
```

```
{ "result" : "test_map_reduce", "ok" : 1 }
```

```
> show collections
Average_Stars
Subreview
business
review
test_map_reduce
user
```

```
> db.test_map_reduce.find()
{ "_id" : "6zKU06XeCBdRDnzgNB5NmQ", "value" : { "count" : 1, "stars" : 2.5, "avg" : 2.5 }
}
{ "_id" : "a8RfCOr41Jgp7urWbKSAzg", "value" : { "count" : 1, "stars" : 2.5, "avg" : 2.5 }
}
{ "_id" : "ZpPd9a-Tv_NGXh8P1ivjzA", "value" : { "count" : 1, "stars" : 2.5, "avg" : 2.5 }
}
{ "_id" : "Jvx8CyM8om6tZvn8PXOWEA", "value" : { "count" : 1, "stars" : 3.5, "avg" : 3.5 }
}
{ "_id" : "O7uzVmRB8GDMU8XaX5l7Eg", "value" : { "count" : 1, "stars" : 3, "avg" : 3 } }
{ "_id" : "1-QYao7fyZ0ihnM16DN9hw", "value" : { "count" : 1, "stars" : 4.5, "avg" : 4.5 }
}
{ "_id" : "6a6IaS9T4uLUR6s7ak3hJQ", "value" : { "count" : 1, "stars" : 5, "avg" : 5 } }
{ "_id" : "kp9o9uv_scnT6ZBdDvvLaA", "value" : { "count" : 1, "stars" : 4, "avg" : 4 } }
{ "_id" : "BCORM4dEleJXnyPXoaH3HQ", "value" : { "count" : 1, "stars" : 4, "avg" : 4 } }
{ "_id" : "u09bcA8mmWEAusKBudAidg", "value" : { "count" : 1, "stars" : 4.5, "avg" : 4.5 }
}
{ "_id" : "w_0nQYHhXhrSR95pWZafVg", "value" : { "count" : 1, "stars" : 4, "avg" : 4 } }
{ "_id" : "r6tNDARvVhHcRD88DEND0Q", "value" : { "count" : 1, "stars" : 4, "avg" : 4 } }
{ "_id" : "mVDXeHkpWK-GNqdi7fN4MQ", "value" : { "count" : 1, "stars" : 3, "avg" : 3 } }
{ "_id" : "mA_XSJGBwGm9RgHndazwiw", "value" : { "count" : 1, "stars" : 3, "avg" : 3 } }
{ "_id" : "xi4yGWRFBRaLDygBj2lqBw", "value" : { "count" : 1, "stars" : 2.5, "avg" : 2.5 }
}
{ "_id" : "laac2uH1lQVzBjKFUjuA1Q", "value" : { "count" : 1, "stars" : 3, "avg" : 3 } }
{ "_id" : "0jAexNHRHhkQv4zNYqenBA", "value" : { "count" : 1, "stars" : 3, "avg" : 3 } }
{ "_id" : "MAKV_ZbhTcVA-IcdjKYhRQ", "value" : { "count" : 1, "stars" : 2.5, "avg" : 2.5 }
}
{ "_id" : "RhGZ1T3d8hodfvuZER7pIg", "value" : { "count" : 1, "stars" : 5, "avg" : 5 } }
{ "_id" : "kQIFaTQW1Q4Lc8ksvyT9QQ", "value" : { "count" : 1, "stars" : 4, "avg" : 4 } }
Type "it" for more
```

Lab 3

1.

MATCH (city:CityNode)
RETURN city
LIMIT 10

```
yelp$ MATCH (city:CityNode) RETURN city LIMIT 10
```

Graph

Table

Text

Code

"city"
{"city":"Moon Township"}
{"city":"montreal"}
{"city":"Uxbridge"}
{"city":"Warrensville Hts."}
{"city":"Warrensville"}
{"city":"Phoniex"}
{"city":"Norton"}
{"city":"Ahwatukee"}
{"city":"Eastlake"}
{"city":"Huntingdon"}

2.

MATCH (business:BusinessNode {city:'Ambridge'}
RETURN business

```
yelp$ MATCH (business:BusinessNode {city:'Ambridge'}) RETURN business
```

Graph
Table
Text
Code

"business"
{"reviewcount":"3","address":"400 Merchant St","city":"Ambridge","latitude":"40.5840997","businessid":"dJ0R-XT78LUQeNHQkD-G9g","name":"Ice Cream Therapy","stars":"3.5","longitude":"-80.225135"}
{"reviewcount":"4","address":"304 Duss Ave","city":"Ambridge","latitude":"40.5823797","businessid":"3gL18eXylqutlzqb6TmB0w","name":"Action Tire Company","stars":"5.0","longitude":"-80.2238749"}
{"reviewcount":"18","address":"603 Duss Ave","city":"Ambridge","latitude":"40.5879393","businessid":"Q_0eGl-aElqHKukHvmLdwA","name":"Nelia's Smokehouse","stars":"4.0","longitude":"-80.2248851"}
{"reviewcount":"14","address":"755 Merchant St","city":"Ambridge","latitude":"40.5886057","businessid":"Eu_zPTrNVAXkpdSxf7CJ2w","name":"K & N Restaurant","stars":"4.5","longitude":"-80.2291033"}
{"reviewcount":"4","address":"663 Merchant St","city":"Ambridge","latitude":"40.5876498","businessid":"Yjf0i2J9q52dYIT8UVGT3g","name":"Heritage Floral Shoppe","stars":"4.5","longitude":"-80.2284934"}
{"reviewcount":"5","address":"598 Merchant St","city":"Ambridge","latitude":"40.5865326","businessid":"y3IVqEFHmrkgVKj2×1Ci4w","name":"Off the Hook Exotics","stars":"4.0","longitude":"-80.2269877"}
{"reviewcount":"32","address":"1007 Merchant St","city":"Ambridge","latitude":"40.592077","businessid":"729grSa1Wsn-hfv7D5uOxg","name":"Pizza House","stars":"4.5","longitude":"-80.230377"}

3.

MATCH (r:ReviewNode {reviewid:'rEITo90tpyKmEfNDp3Ou3A'})-[:Reviewed]->(business:BusinessNode)
RETURN business

```
yelp$ MATCH (r:ReviewNode {reviewid:'rEITo90tpyKmEfNDp3Ou3A'})-[:Reviewed]→(business:BusinessNode) RETURN business
```

Graph
Table
Text
Code

"business"
{"reviewcount":"106","address":"2800 E Germann Rd","city":"Chandler","latitude":"33.278104","businessid":"6lj2BJ4tJeu7db5asGHQ4w","name":"Slim Chickens","stars":"3.0","longitude":"-111.7924641"}

4.

MATCH (user:UserNode)-[:Review]->(:ReviewNode)-[:Reviewed]->(:BusinessNode {businessid:'fyJAqmweGm8VXnpU4CWGNw'})
RETURN user.name, user.fans

```
yelp$ MATCH (user:UserNode)-[:Review]→(:Rev
```

Table

Text

Code

"user.name"	"user.fans"
"Joseph"	"0"
"Linda"	"0"
"Brittanie"	"24"
"Melinda"	"17"
"Kellie"	"1"
"Calinda"	"0"
"Xandon"	"0"
"Teresa"	"78"
"Island"	"1"
"John"	"2"
"Chris"	"0"
"Hicks"	"0"
"Lucy"	"0"
"Shawna"	"1"
"Rochelle"	"0"

5.

MATCH (:UserNode {userid:'TEtzbpgA2BFBrC0y0sCbfw'})-[:Review]->(:ReviewNode {stars:'5.0'})-[:Reviewed]->(business:BusinessNode)
RETURN business.name, business.address

```
yelp$ MATCH (:UserNode {userid:'TEtzbpgA2BFBrC0y0sCbfw'})-[:Review]→(:ReviewNode {stars:'5.0'})-[:Reviewed]→(business:Busines…
```

Table | Text | Code

	business.name	business.address
1	"The Buffet"	"3131 Las Vegas Blvd S"
2	"MGM Grand Hotel"	"3799 Las Vegas Blvd S"
3	"Paris Las Vegas Hotel & Casino"	"3655 Las Vegas Blvd S"
4	"Blue Man Group"	"3900 Las Vegas Blvd S"
5	"Burger Bar"	"3930 S Las Vegas Blvd, Ste 121A"
6	"Bellagio Hotel"	"3600 S Las Vegas Blvd"
7	"The Venetian Las Vegas"	"3355 South Las Vegas Boulevard"

6.

MATCH (business:BusinessNode)
RETURN business.name, business.stars, business.address
ORDER BY business.stars DESC
LIMIT 15

```
yelp$ MATCH (business:BusinessNode) RETURN business.name, business.stars, business.address ORDER BY business.stars DESC LIMIT 15
```

Table | Text | Code

	business.name	business.stars	business.address
1	"Wesley Friedman - FBC Mortgage"	"5.0"	"6775 Edmond St, Ste 210"
2	"Stephen L Walker, DDS, MS"	"5.0"	"2220 W Southern Ave, Ste 102, Endodontic Specialists Ltd"
3	"Knot Salon"	"5.0"	"4848 E Cactus Rd, Ste 100"
4	"Brian's Furniture"	"5.0"	"30808 Center Ridge Rd"
5	"Three Square"	"5.0"	"4190 N Pecos Rd"
6	"Key To Healing Massage"	"5.0"	"14202 N Scottsdale Rd"
7	"Farmers Insurance - Paul Lorenz"	"5.0"	"15655 W Roosevelt St, Ste 237"
8	"Vita Bella Fine Day Spa"	"5.0"	"5940 W Union Hills Dr"
9	"Maurices"	"5.0"	"7981 W Tropical Pkwy, Ste 120"
10	"OzBraz"	"5.0"	"560 E Germann Rd, Ste 107"
11	"Myron Hensel Photography"	"5.0"	""

Started streaming 15 records after 1 ms and completed after 161 ms.

7.

MATCH (user:UserNode)
WHERE tointeger(user.fans)>200
RETURN user.name, user.fans
LIMIT 10

```
yelp$ MATCH (user:UserNode) WHERE tointeger(user.fans)>200 RETURN user.name, user.fans LIMIT 10
```

Table Text Code

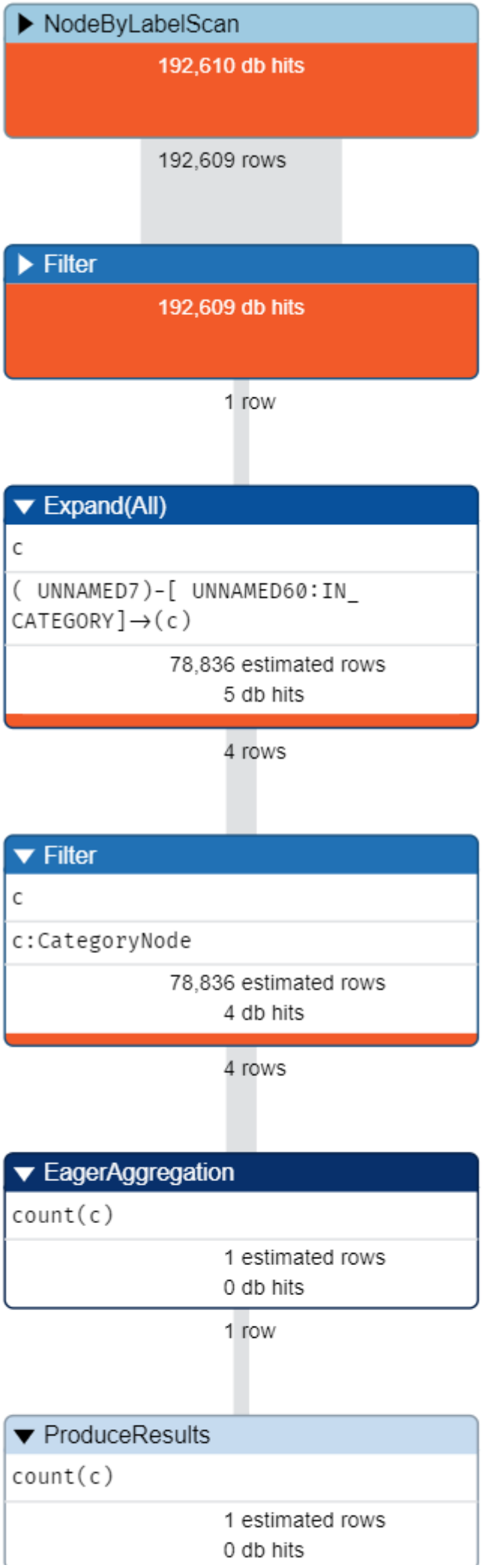
	user.name	user.fans
1	"Keane"	"696"
2	"Annie"	"407"
3	"Jason"	"251"
4	"Regina"	"231"
5	"Alma"	"272"
6	"Stephanie"	"282"
7	"Michael"	"912"
8	"Nette"	"426"
9	"Alan"	"379"
10	"Keila"	"447"

8.

MATCH (:BusinessNode {businessid:'tyjquHslrAuF5EUejbPfrw'})-[:IN_CATEGORY]->(c:CategoryNode)
RETURN count(c)

使用PROFILE查看执行计划：

PROFILE MATCH (:BusinessNode {businessid:'tyjquHslrAuF5EUejbPfrw'})-[:IN_CATEGORY]->(c:CategoryNode)
RETURN count(c)


NodeByLabelScan
192,610 db hits
192,609 rows
Filter
192,609 db hits
1 row
Expand(All)
c
(UNNAMED7)-[UNNAMED60:IN_CATEGORY]→(c)
78,836 estimated rows
5 db hits
4 rows
Filter
c
c:CategoryNode
78,836 estimated rows
4 db hits
4 rows
EagerAggregation
count(c)
1 estimated rows
0 db hits
1 row
ProduceResults
count(c)
1 estimated rows
0 db hits

9.

1 row

MATCH (business:BusinessNode {businessid:'tyjquHslrAuF5EUejbPfrw'})-[:IN_CATEGORY]->(c:CategoryNode)
RETURN collect(c.category) as category

Result

	category
1	[" Latin American", " Venezuelan", " Restaurants", "Mexican"]

10.

MATCH(:UserNode {name:'Allison'})-[:HasFriend]->(friend:UserNode)
WITH friend.name as friendsList, size((friend) -[:HasFriend]->()) as numofFOFs
RETURN friendsList, numofFOFs

Table
Text
Code

	friendsList	numofFOFs
1	"David"	6
2	"Regina"	32
3	"Michael"	292
4	"Greg"	73
5	"Jennifer"	375
6	"Jerry"	85
7	[illegible]	[illegible]

Started streaming 19478 records in less than 1 ms and completed after 53 ms, displaying first 1000 rows.

11.

MATCH (business:BusinessNode)-[:IN_CATEGORY]->(:CategoryNode {category:'Salad'})
WITH business.city as city, count(*) as count
RETURN city, count
ORDER BY count DESC
LIMIT 5

	city	count
1	"Toronto"	52
2	"Charlotte"	40
3	"Phoenix"	39
4	"Las Vegas"	39
5	"Pittsburgh"	19

12.

```
MATCH (business:BusinessNode)
WITH business.name as name, count(*) as cnt
RETURN name, cnt
ORDER BY cnt DESC
LIMIT 10
```

	name	cnt
1	"Starbucks"	1066
2	"McDonald's"	806
3	"Subway"	768
4	"Tim Hortons"	333
5	"Pizza Hut"	320
6	"Taco Bell"	313
7	"Burger King"	302
8	"Walgreens"	301
9	"Wendy's"	294
10	"The UPS Store"	279

13.

```
MATCH (businessall:BusinessNode)
WITH count(distinct businessall.name) as cnt
MATCH (business:BusinessNode)
WHERE tointeger(business.reviewcount) > 5000
WITH cnt, count(business.name) as subcnt, business.name as name, business.reviewcount as
reviewcount
RETURN subcnt*1.0/cnt, name, reviewcount
ORDER BY reviewcount DESC
```

	subcnt*1.0/cnt	name	reviewcount
1	0.00000689436454641975 7	"Mon Ami Gabi"	"8348"
2	0.00000689436454641975 7	"Bacchanal Buffet"	"8339"
3	0.00000689436454641975 7	"Wicked Spoon"	"6708"
4	0.00000689436454641975 7	"Hash House A Go Go"	"5763"
5	0.00000689436454641975 7	"Gordon Ramsay BurGR"	"5484"
6	0.00000689436454641975 7	"Earl of Sandwich"	"5075"

14.

```
MATCH (business:BusinessNode)-[:IN_CATEGORY]->(:CategoryNode {category:'Zoos'})
WITH business
MATCH (:ReviewNode {stars:'5.0'})-[:Reviewed]->(business:BusinessNode)
RETURN distinct business.city
```

	business.city
1	"Toronto"
2	"Cave Creek"
3	"Las Vegas"
4	"Calgary"

15.

```
MATCH (user:UserNode)-[:Review]->(:ReviewNode)-[:Reviewed]->(business:BusinessNode)
WITH count(distinct user) as user_count, business
RETURN business.businessid, business.name, user_count
ORDER BY user_count DESC
LIMIT 10
```

	business.businessid	business.name	user_count
1	"4JNXUYY8wbaaDmk3BPzlWw"	"Mon Ami Gabi"	8349
2	"RESDUcs7fliihp38-d6_6g"	"Bacchanal Buffet"	8341
3	"K7lWdNUhCbcnEvI0NhGewg"	"Wicked Spoon"	6710
4	"f4x1YBxkLrZg652xt2KR5g"	"Hash House A Go Go"	5763
5	"cYwJA2A6I12KNkm2rtXd5g"	"Gordon Ramsay BurGR"	5484
6	"DkYS3arLOhA8si5uUEmHOw"	"Earl of Sandwich"	5076
7	"2weQS-RnoOBhb1KsHKyoSQ"	"The Buffet"	4401
8	"5LNZ 67Yw9RD6nf4_UhXOjw"	"The Cosmopolitan of Las Vegas"	4322
9	"iCQpiavjjPzJ5_3gPD5Ebg"	"Secret Pizza"	4286
10	"ujHiaprwCQ5ewziu0Vi9rw"	"The Buffet at Bellagio"	4230

16.

准备工作，使用语句为UserNode新建一个属性flag用来实验

MATCH (user:UserNode)

SET user.flag = user.fans

ERROR **ServiceUnavailable**

```
Could not perform discovery. No routing servers available. Known routing table: RoutingTable[database=yelp,
expirationTime=1698241200886, currentTime=1698241027220, routers=[], readers=[], writers=[]]
```

```
OpenJDK 64-Bit Server VM warning: INFO: os::commit_memory(0x00000000ffa00000, 1048576, 0) failed; error='Not enough space' (errno=12)
#
# There is insufficient memory for the Java Runtime Environment to continue.
# Native memory allocation (mmap) failed to map 1048576 bytes for committing reserved memory.
# An error report file with more information is saved as:
# /root/neo4j-community-4.0.9/bin/hs_err_pid1956.log
```

由于服务器内存不够而报错。因此需要对属性作用的范围做出一定的限定，例如fans大于400，于是属性创建完成

MATCH (user:UserNode)

WHERE toInteger(user.fans) > 400

SET user.flag = user.fans

Set 295 properties, completed after 1617 ms.

对UserNode的flag属性执行查询（flag>8000）、创建（flag=10000）、更新（flag=456 to flag=3456）、删除（flag>4000）操作

查询：

MATCH (user:UserNode)
WHERE toInteger(user.flag) > 8000
RETURN user

"user"
{"flag":"9538","cool":"13227","name":"Mike","userid":"37cpUoM8hlkSQfReIEBd-Q","useful":"19715","funny":"10085","fans":"9538"}

Started streaming 1 records after 2 ms and completed after 713 ms.

创建：

```
MATCH (user:UserNode)
WHERE toInteger(user.fans) = 198
SET user.flag = 10000
```

Set 8 properties, completed after 643 ms.

更新：

```
MATCH (user:UserNode)
WHERE toInteger(user.flag) = 456
SET user.flag = 3456
```

Completed after 495 ms.

删除：

```
MATCH (user:UserNode)
WHERE user.flag > 4000
REMOVE user.flag
```

Set 8 properties, completed after 12 ms.

重置属性信息（最上面的步骤），然后为UserNode的属性flag建立索引

```
CREATE INDEX FOR (user:UserNode) ON (user.flag)
```

Added 1 index, completed after 15 ms.

对UserNode的flag属性执行相同的查询、创建、更新、删除操作，比较操作时间

查询：

Started streaming 1 records in less than 1 ms and completed after 507 ms.

创建：

Set 8 properties, completed after 731 ms.

更新：

Set 1 property, completed after 584 ms.

删除：

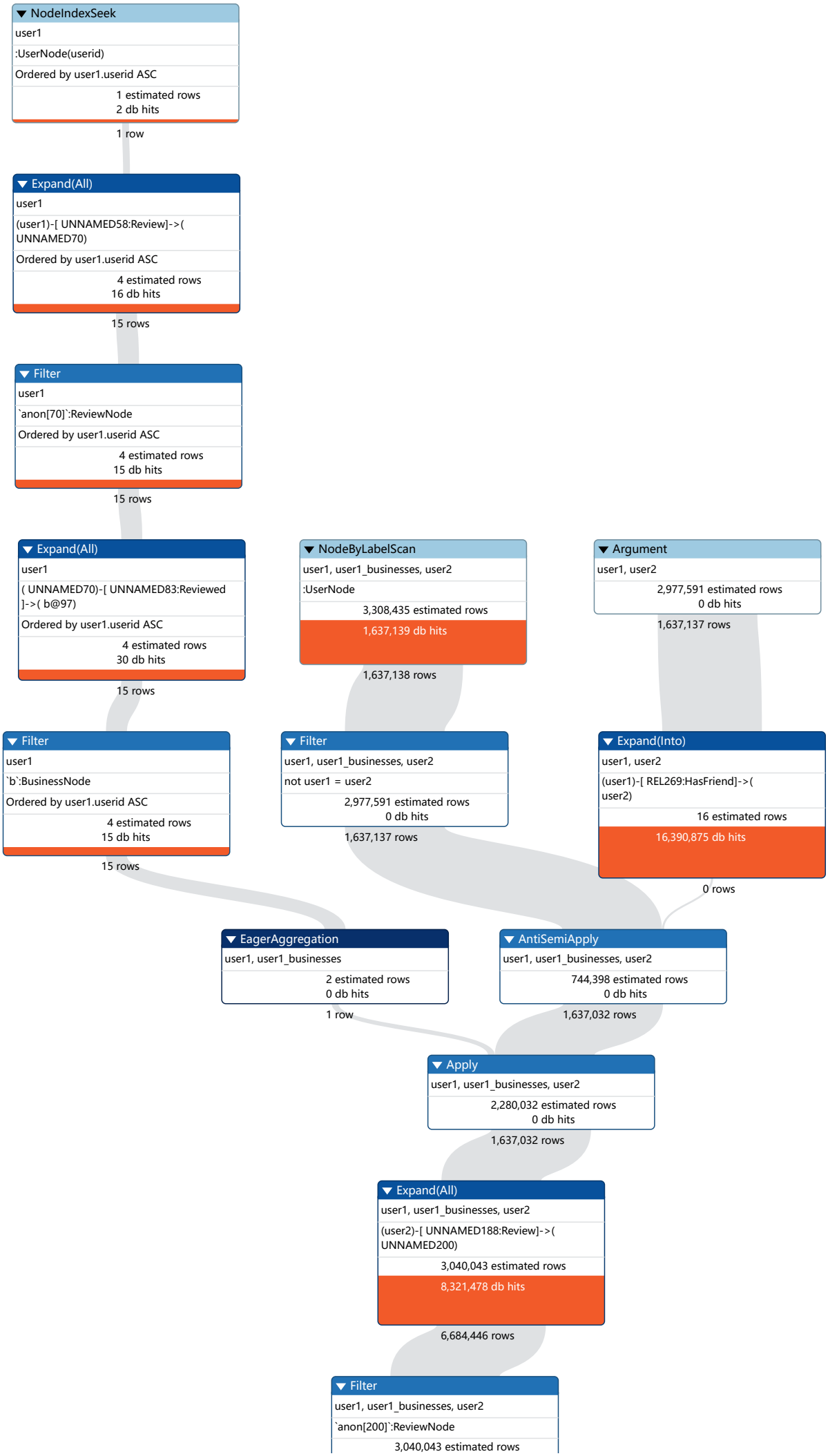
Set 11 properties, completed after 673 ms.

17.

```
MATCH (user1:UserNode {userid: 'tvZKPah2u9G9dFBg5GT0eg'})-[:Review]->(:ReviewNode)-[:Reviewed]->(b:BusinessNode)
WITH user1, COLLECT(DISTINCT b) AS user1_businesses
MATCH (user2:UserNode)-[:Review]->(:ReviewNode)-[:Reviewed]->(b:BusinessNode)
WHERE NOT EXISTS ((user1)-[:HasFriend]->(user2)) AND b IN user1_businesses AND user1<>user2
WITH user1, user2, COUNT(DISTINCT b) AS num
RETURN user1.name, user2.name, num
ORDER BY num DESC
```

Table Text Code

	user1.name	user2.name	num
1	"Emily"	"Anthony"	10
2	"Emily"	"Norm"	10
3	"Emily"	"Michael"	8
4	"Emily"	"Rodney"	8
5	"Emily"	"Janice"	8
6	"Emily"	"Lorrie"	7
7	"Emily"	"Darrell"	7
8	"Emily"	"J"	7


NodeIndexSeek
user1
:UserNode(userid)
Ordered by user1.userid ASC
1 estimated rows
2 db hits
1 row
Expand(All)
user1
(user1)-[UNNAMED58:Review]->(UNNAMED70)
Ordered by user1.userid ASC
4 estimated rows
16 db hits
15 rows
Filter
user1
`anon[70]`:ReviewNode
Ordered by user1.userid ASC
4 estimated rows
15 db hits
15 rows
Expand(All)
user1
(UNNAMED70)-[UNNAMED83:Reviewed]->(b@97)
Ordered by user1.userid ASC
4 estimated rows
30 db hits
15 rows
NodeByLabelScan
user1, user1_businesses, user2
:UserNode
3,308,435 estimated rows
1,637,139 db hits
1,637,138 rows
Argument
user1, user2
2,977,591 estimated rows
0 db hits
1,637,137 rows
Filter
user1
`b`:BusinessNode
Ordered by user1.userid ASC
4 estimated rows
15 db hits
15 rows
Filter
user1, user1_businesses, user2
not user1 = user2
2,977,591 estimated rows
0 db hits
1,637,137 rows
Expand(Into)
user1, user2
(user1)-[REL269:HasFriend]->(user2)
16 estimated rows
16,390,875 db hits
0 rows
EagerAggregation
user1, user1_businesses
2 estimated rows
0 db hits
1 row
AntiSemiApply
user1, user1_businesses, user2
744,398 estimated rows
0 db hits
1,637,032 rows
Apply
user1, user1_businesses, user2
2,280,032 estimated rows
0 db hits
1,637,032 rows
Expand(All)
user1, user1_businesses, user2
(user2)-[UNNAMED188:Review]->(UNNAMED200)
3,040,043 estimated rows
8,321,478 db hits
6,684,446 rows
Filter
user1, user1_businesses, user2
`anon[200]`:ReviewNode
3,040,043 estimated rows

Started streaming 16061 records after 50 ms and completed after 68 ms

优化：

首先，可以为userid和businessid建立索引

CREATE INDEX FOR (user:UserNode) ON (user.userid)

CREATE INDEX FOR (b:BusinessNode) ON (b.businessid)

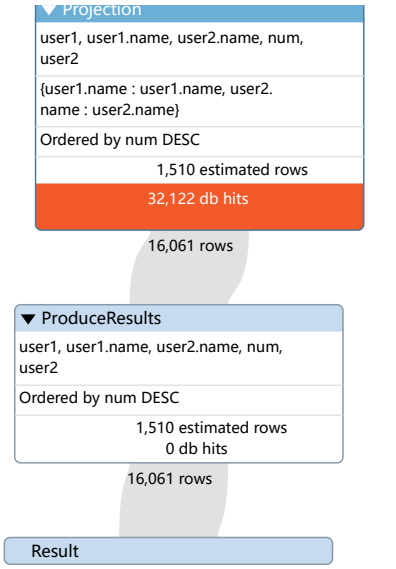
Added 1 index, completed after 26 ms.

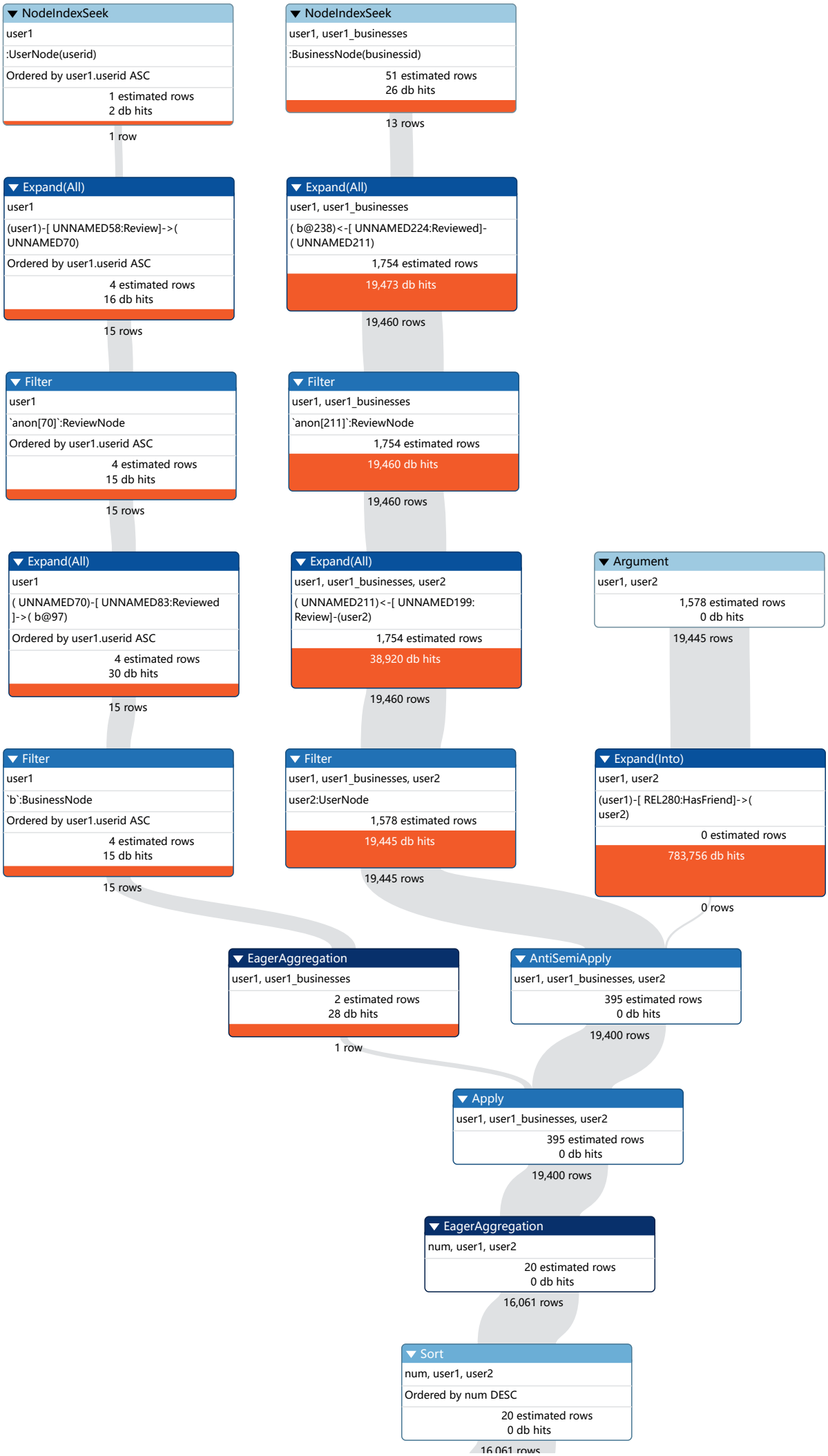
Added 1 index, completed after 4 ms.

然后将上面的COLLECT函数改为仅对business.businessid聚合，减少内存开销

```
MATCH (user1:UserNode {userid: 'tvZKPah2u9G9dFBg5GT0eg'})-[:Review]->(:ReviewNode)-[:Reviewed]->(b:BusinessNode)
WITH user1, COLLECT(DISTINCT b.businessid) AS user1_businesses
MATCH (user2:UserNode)-[:Review]->(:ReviewNode)-[:Reviewed]->(b:BusinessNode)
WHERE NOT EXISTS ((user1)-[:HasFriend]->(user2)) AND b.businessid IN user1_businesses AND user1<>user2
WITH user1, user2, COUNT(DISTINCT b) AS num
RETURN user1.name, user2.name, num
ORDER BY num DESC
```

Started streaming 4904 records after 1 ms and completed after 1 ms, displaying first 1000 rows.

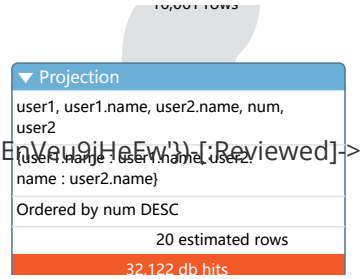


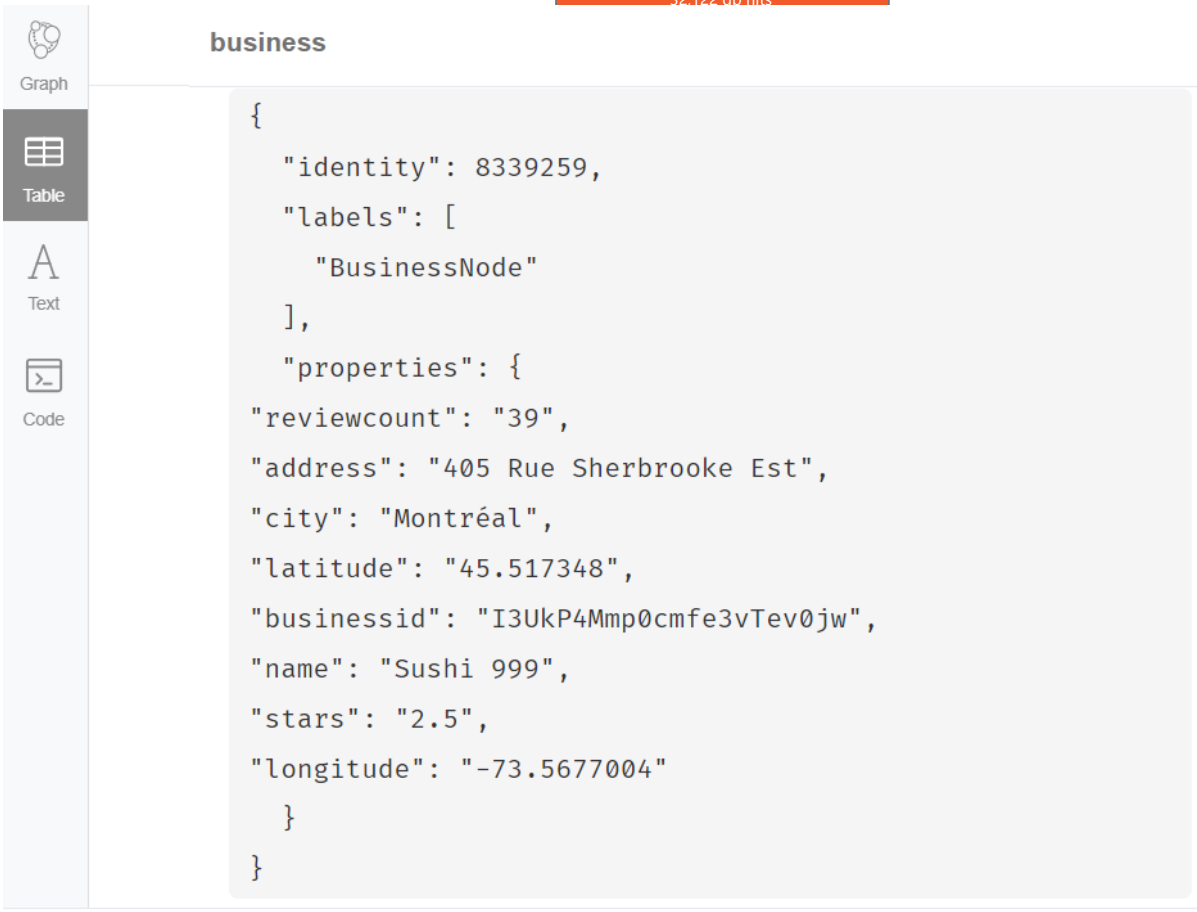

NodeIndexSeek
user1
:UserNode(userid)
Ordered by user1.userid ASC
1 estimated rows
2 db hits
1 row
NodeIndexSeek
user1, user1_businesses
:BusinessNode(businessid)
51 estimated rows
26 db hits
13 rows
Expand(All)
user1
(user1)-[UNNAMED58:Review]->(UNNAMED70)
Ordered by user1.userid ASC
4 estimated rows
16 db hits
15 rows
Expand(All)
user1, user1_businesses
(b@238)<-[UNNAMED224:Reviewed]- (UNNAMED211)
1,754 estimated rows
19,473 db hits
19,460 rows
Filter
user1
`anon[70]`:ReviewNode
Ordered by user1.userid ASC
4 estimated rows
15 db hits
15 rows
Filter
user1, user1_businesses
`anon[211]`:ReviewNode
1,754 estimated rows
19,460 db hits
19,460 rows
Expand(All)
user1
(UNNAMED70)-[UNNAMED83:Reviewed]->(b@97)
Ordered by user1.userid ASC
4 estimated rows
30 db hits
15 rows
Expand(All)
user1, user1_businesses, user2
(UNNAMED211)<-[UNNAMED199: Review]-(user2)
1,754 estimated rows
38,920 db hits
19,460 rows
Argument
user1, user2
1,578 estimated rows
0 db hits
19,445 rows
Filter
user1
`b`:BusinessNode
Ordered by user1.userid ASC
4 estimated rows
15 db hits
15 rows
Filter
user1, user1_businesses, user2
user2:UserNode
1,578 estimated rows
19,445 db hits
19,445 rows
Expand(Into)
user1, user2
(user1)-[REL280:HasFriend]->(user2)
0 estimated rows
783,756 db hits
0 rows
EagerAggregation
user1, user1_businesses
2 estimated rows
28 db hits
1 row
AntiSemiApply
user1, user1_businesses, user2
395 estimated rows
0 db hits
19,400 rows
Apply
user1, user1_businesses, user2
395 estimated rows
0 db hits
19,400 rows
EagerAggregation
num, user1, user2
20 estimated rows
0 db hits
16,061 rows
Sort
num, user1, user2
Ordered by num DESC
20 estimated rows
0 db hits
16,061 rows

18.

Neo4j查询：

MATCH (:ReviewNode {reviewid:'TIYgnDzezfeEnVeu9jHeEw'})-[:Reviewed]->(business:BusinessNode)
RETURN business






```
{
  "identity": 8339259,
  "labels": [
    "BusinessNode"
  ],
  "properties": {
"reviewcount": "39",
"address": "405 Rue Sherbrooke Est",
"city": "Montréal",
"latitude": "45.517348",
"businessid": "I3UkP4Mmp0cmfe3vTev0jw",
"name": "Sushi 999",
"stars": "2.5",
"longitude": "-73.5677004"
  }
}
```


Started streaming 1 records after 1 ms and completed after 18146 ms.

MongoDB查询：

var bid=db.review.findOne({"review_id":"TIYgnDzezfeEnVeu9jHeEw"}).business_id

db.business.findOne({"business_id":bid})

```
> var bid=db.review.findOne({"review_id":"TIYgnDzezfeEnVeu9jHeEw"}).business_id
> db.business.findOne({"business_id":bid})
{
        "_id" : ObjectId("6016c6b5af81085b0f21b190"),
        "business_id" : "I3UkP4Mmp0cmfe3vTev0jw",
        "name" : "Sushi 999",
        "address" : "405 Rue Sherbrooke Est",
        "city" : "Montréal",
        "state" : "QC",
        "postal_code" : "H2L 1J9",
        "latitude" : 45.517348,
        "longitude" : -73.5677004,
        "stars" : 2.5,
        "review_count" : 39,
        "is_open" : 0,
        "attributes" : {
                "RestaurantsGoodForGroups" : "True",
                "BikeParking" : "False",
                "NoiseLevel" : "u'average'",
                "GoodForKids" : "True",
                "WiFi" : "u'free'",
                "BusinessParking" : "{'garage': False, 'street': True, 'validated': False,
 'lot': False, 'valet': False}",
                "RestaurantsDelivery" : "True",
                "RestaurantsReservations" : "True",
                "Alcohol" : "'beer_and_wine'",
                "HasTV" : "True",
                "RestaurantsPriceRange2" : "2",
                "OutdoorSeating" : "False",
                "RestaurantsTakeOut" : "True",
                "RestaurantsAttire" : "u'casual'",
                "Ambience" : "{'romantic': False, 'intimate': False, 'classy': False, 'hip
ster': False, 'divey': False, 'touristy': False, 'trendy': False, 'upscale': False, 'casua
l': True}"
        },
        "categories" : [
                "Sushi Bars",
                "Japanese",
                "Restaurants"
        ],
        "hours" : null,
        "loc" : {
                "type" : "Point",
                "coordinates" : [
                        -73.5677004,
                        45.517348
                ]
        }
}
```

MongoDB查询所用时间略大于Neo4j，据此可以指出Neo4j和MongoDB主要的适用场景。

Neo4j适用于复杂的图结构数据查询，如社交网络、推荐系统、知识图谱等。由于其使用基于图论的查询语言Cypher，支持关系型数据的快速查询和分析。

而MongoDB适用于海量非结构化或半结构化数据存储和查询，如日志、传感器数据、文档数据库等。由于其使用文档数据库模型，支持高效的数据插入和查询，以及分布式数据库集群的横向扩展。

Lab 4

1.

MATCH (user:UserNode)-[:Review]->(:ReviewNode)-[:Reviewed]->(business:BusinessNode)
WITH user, COUNT(distinct business) as count
WHERE count>5
RETURN user.name, user.funny, user.fans, count

224936	Crystal	0	0		
224937					
224938					

2.

将1得到的结果导入MongoDB，并使用该表格数据，统计其中所有出现的用户名及该用户名对应的出现次数，并按照出现次数降序排序,使用aggregate实现

1）从Neo4j的查询中导出csv文件（export.csv）

2）在mongodb新建集合from_neo4j，将csv文件导入集合

C:\GAP\大数据管理实验>scp ./export.csv root@1.94.55.43:/root/

```
C:\GAP\大数据管理实验>scp ./export.csv root@1.94.55.43:/root/
The authenticity of host '1.94.55.43 (1.94.55.43)' can't be established.
ED25519 key fingerprint is SHA256:HO4FMGJ9ay3Di2cnNBp/IZ3bkotVikwwN8BWJLP4NR8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added '1.94.55.43' (ED25519) to the list of known hosts.
root@1.94.55.43's password:
export.csv                                                    100% 2375KB   8.7MB/s   00:00
```

```
root@ecs-c925:~# ls
data  export.csv  mysql-apt-config_0.8.10-1_all.deb
```

然后启动mongo，选择yelp数据库，创建一个新的集合from_neo4j

db.createCollection("from_neo4j")

```
> db.createCollection("from_neo4j")
{ "ok" : 1 }
```

使用show collections查看当前集合：

```
> show collections
Average_Stars
Subreview
business
from_neo4j
review
test_map_reduce
user
```

退出mongoDB，回到主目录，把数据导入到mongoDB中yelp数据集的from_neo4j集合中。

mongoimport -d=yelp -c=from_neo4j --type=csv --headerline ./export.csv

```
root@ecs-c925:~# mongoimport -d=yelp -c=from_neo4j --type=csv --headerline ./export.csv
2023-10-27T11:35:41.800+0800    connected to: mongodb://localhost/
2023-10-27T11:35:43.261+0800    224935 document(s) imported successfully. 0 document(s) failed to import.
```

3）统计其中所有出现的用户名及该用户名对应的出现次数，并按照出现次数降序排序。

```
db.from_neo4j.aggregate([
{$group:{_id:'$u.name', count:{$sum:1}}},
{$sort:{count: -1}}
])
```

```
{ "_id" : "John", "count" : 1864 }
{ "_id" : "Michael", "count" : 1804 }
{ "_id" : "David", "count" : 1736 }
{ "_id" : "Chris", "count" : 1687 }
{ "_id" : "Jennifer", "count" : 1649 }
{ "_id" : "Mike", "count" : 1571 }
{ "_id" : "Jessica", "count" : 1469 }
{ "_id" : "Sarah", "count" : 1346 }
{ "_id" : "Michelle", "count" : 1333 }
{ "_id" : "Lisa", "count" : 1204 }
{ "_id" : "Jason", "count" : 1101 }
{ "_id" : "Mark", "count" : 1088 }
{ "_id" : "Ashley", "count" : 1083 }
{ "_id" : "Amy", "count" : 1017 }
{ "_id" : "Amanda", "count" : 1013 }
{ "_id" : "Stephanie", "count" : 1003 }
{ "_id" : "Brian", "count" : 993 }
{ "_id" : "J", "count" : 986 }
{ "_id" : "Melissa", "count" : 953 }
{ "_id" : "Nicole", "count" : 943 }
Type "it" for more
```

3.

MATCH (business:BusinessNode)-[:IN_CATEGORY]->(c:CategoryNode)
RETURN business.name as name, business.city as category

	name	city	category
1	"Rohaley's Auto & Truck Repair"	"Mentor"	"Commercial Truck Repair"
2	"AAA Muffler"	"Henderson"	"Commercial Truck Repair"
3	"Parkway Auto Care"	"Berea"	"Commercial Truck Repair"
4	"Master Tech Collision"	"Tempe"	"Commercial Truck Repair"
5	"Regal Truck & Trailer Repair"	"Mississauga"	"Commercial Truck Repair"
6	"Aone Mobile Mechanics"	"Las Vegas"	"Commercial Truck Repair"
7	"Diversified Truck & Equipment Sales Inc"	"Mesa"	"Commercial Truck Repair"
8	"Sizzling Hut"	"DeForest"	"Szechuan"
9	"Spicy City"	"Las Vegas"	"Szechuan"
10	"China Star"	"Pittsburgh"	"Szechuan"
11	"Tasty China"	"North Las Vegas"	"Szechuan"

Started streaming 788359 records after 1 ms and completed after 3 ms, displaying first 1000 rows.

然后类似于上面第2题的步骤：

```
root@ecs-c925:~# ls
AllBusiness.csv  data
```

db.createCollection("AllBusiness")

```
> db.createCollection("AllBusiness")
{ "ok" : 1 }
```

退出mongoDB，回到主目录，把数据导入到mongoDB中yelp数据集的AllBusiness集合中。

mongoimport -d=yelp -c=AllBusiness --type=csv --headerline ./AllBusiness.csv

```
root@ecs-c925:~# mongoimport -d=yelp -c=AllBusiness --type=csv --headerline ./AllBusiness.csv
2023-10-27T15:40:02.973+0800    connected to: mongodb://localhost/
2023-10-27T15:40:05.973+0800    [###########.............] yelp.AllBusiness     16.1MB/32.3MB (50.0%)
2023-10-27T15:40:08.973+0800    [########################.] yelp.AllBusiness     31.7MB/32.3MB (98.0%)
2023-10-27T15:40:09.090+0800    [########################.] yelp.AllBusiness     32.3MB/32.3MB (100.0%)
2023-10-27T15:40:09.090+0800    788359 document(s) imported successfully. 0 document(s) failed to import.
```

接下来使用aggregate对AllBusiness去重，仅保留城市、商铺类型。首先创建一个集合用于保存结果

db.createCollection("DistinctBusiness")

db.AllBusiness.aggregate([{ $group: { *id: { city: '$city', category: '$category' } } }]).forEach((item) => { *db.DistinctBusiness.insert(item.*id) })

查看结果：

```
> db.DistinctBusiness.count()
67536
> db.DistinctBusiness.find().limit(5)
{ "_id" : ObjectId("653b6a55c84e3a41a5adf21b"), "city" : "Brecksville", "category" : "Fitness & Instruction"
{ "_id" : ObjectId("653b6a55c84e3a41a5adf21c"), "city" : "Phoenix", "category" : "Elementary Schools" }
{ "_id" : ObjectId("653b6a55c84e3a41a5adf21d"), "city" : "Oberlin", "category" : "Pets" }
{ "_id" : ObjectId("653b6a55c84e3a41a5adf21e"), "city" : "Goodyear", "category" : "Driving Schools" }
{ "_id" : ObjectId("653b6a55c84e3a41a5adf21f"), "city" : "Markham", "category" : "Building Supplies" }
```

将结果导出到服务器主目录下的result.csv中。

mongoexport -d yelp -c DistinctBusiness --type=csv --fields city,category --out result.csv

然后将其放在neo4j安装目录的import下

cd ~/neo4j-community-4.0.9/import

cp /root/result.csv ./

将去重后的结果导入Neo4j中的新库result中，完成（City-[Has]->Category）图谱的构建。

```
LOAD CSV WITH HEADERS FROM "file:///result.csv" AS f
MERGE (c:CityNode {city: COALESCE(f.city, "")})
MERGE (a:CategoryNode {category: COALESCE(f.category, "")})
CREATE (c) -[:Has]-> (a)
```

Added 125 labels, created 125 nodes, set 125 properties, created 67536 relationships, completed after 234695 ms.

最后查看City-[Has]->Category图谱

MATCH p=()-[r:Has]->()

RETURN p

LIMIT 20

Lab 5

1.

体验 MySQL 在 InnoDB 存储引擎下的 MVCC 多版本并发控制，实现的事务 ACID 特性。请注意 Mysql 需要选用什么事务隔离级来支持 MVCC？请构造多用户多写多读案例来展现 MVCC 并发控制特性，解释各种结果产生的原因。

在MySQL中，首先创建一个数据库testdb

mysql -u root -p

create database testdb;
use testdb;

```
mysql> create database testdb;
Query OK, 1 row affected (0.00 sec)

mysql> use testdb;
Database changed
```

在testdb中创建一个新表，设置engine=InnoDB

```
create table test (
 id int(10) not null,
 name varchar(20) not null,
 flag int(5) not null,
 primary key(id)
) engine=InnoDB;
```

设置事务隔离等级为可重复读（repeatable read）

set session transaction isolation level repeatable read;

插入初始数据：

insert into test VALUES(1, 'LJP', 27);

insert into test VALUES(2, 'DJE', 42);

insert into test VALUES(3, 'OFW', 61);

insert into test VALUES(4, 'SLX', 15);

使用select * from test;查看当前表内的数据：

```
mysql> select * from test
    -> ;
+----+------+------+
| id | name | flag |
+----+------+------+
|  1 | LJP  |   27 |
|  2 | DJE  |   42 |
|  3 | OFW  |   61 |
|  4 | SLX  |   15 |
+----+------+------+
4 rows in set (0.00 sec)
```

开始一个事务：

start transaction;

然后打开会话窗口B（直接在Xshell中完成）。窗口A将id为3的flag更新为88，与此同时，在另外一边，窗口B将id为3的flag更新为99。

start transaction;

update test set flag=88 where id=3; //窗口A

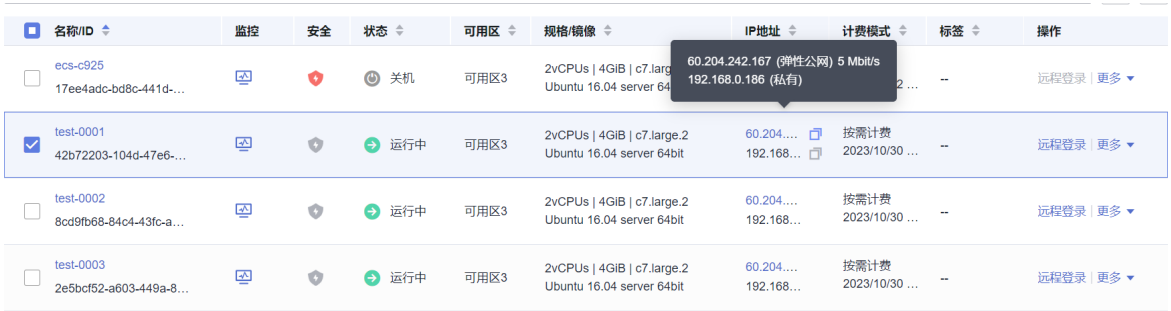
update test set flag=99 where id=3; //窗口B

在窗口A中查询当前表内的数据，可以发现A对id为3的数据更新成功，在本地查询的flag为88。但是对于窗口B，其对id为3的数据的更新操作受到阻塞（“Query execution was interrupted”），且在二次查询时发现该数据的flag仍然为A更新之前的值，也就是说在窗口A对数据的更新操作提交（commit）之前，会屏蔽窗口B对同一数据的更新操作，同时由于MVCC的特性，窗口B在查询数据时仍然会使用本地保存的版本。

现在，再在窗口A中完成提交commit操作，到窗口B中查询当前数据，发现此时的数据变为最新版本。此时窗口B可以正常对id为3的数据进行更新：

2.

体验 MongoDB 的 MVCC，数据集可自建或选用 yelp 数据集中的 test 集合中进行测试，测试方法同 MySQL。请对测试结果进行说明，并与 MySQL 的 MVCC 实验结果进行对比分析。建议创建 MongoDB 副本或分片集群，体验 MVCC 的不同效果（可选做其一）。



创建三台服务器(test-0001 test-0002 test-0003)，弹性公网ip分别为：

60.204.242.167

60.204.228.189

60.204.244.116

添加入方向和出方向规则，检验是否可以ping通：

```
Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xiaoy>ping 60.204.228.189

正在 Ping 60.204.228.189 具有 32 字节的数据:
来自 60.204.228.189 的回复: 字节=32 时间=65ms TTL=46
来自 60.204.228.189 的回复: 字节=32 时间=52ms TTL=46
来自 60.204.228.189 的回复: 字节=32 时间=52ms TTL=46
来自 60.204.228.189 的回复: 字节=32 时间=52ms TTL=46

60.204.228.189 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 52ms，最长 = 65ms，平均 = 55ms
```

三台服务器分别创建数据存放的目录：

```
root@test-0001:~# su - mongodb
No directory, logging in with HOME=/
root@test-0001:~# mkdir /usr/local/mongodb
root@test-0001:~# cd /usr/local/mongodb
root@test-0001:/usr/local/mongodb# mkdir -p data/shard11
root@test-0001:/usr/local/mongodb# mkdir -p data/shard21
root@test-0001:/usr/local/mongodb# mkdir -p data/config
root@test-0001:/usr/local/mongodb# touch data/shard11.log
root@test-0001:/usr/local/mongodb# touch data/shard21.log
root@test-0001:/usr/local/mongodb#
```

```
root@test-0002:~# su - mongodb
No directory, logging in with HOME=/
root@test-0002:~# mkdir /usr/local/mongodb
root@test-0002:~# cd /usr/local/mongodb
root@test-0002:/usr/local/mongodb# mkdir -p data/shard12
root@test-0002:/usr/local/mongodb# mkdir -p data/shard22
root@test-0002:/usr/local/mongodb# mkdir -p data/config
root@test-0002:/usr/local/mongodb# touch data/shard12.log
root@test-0002:/usr/local/mongodb# touch data/shard22.log
root@test-0002:/usr/local/mongodb#
```

```
root@test-0003:~# su - mongodb
No directory, logging in with HOME=/
root@test-0003:~# mkdir /usr/local/mongodb
root@test-0003:~# cd /usr/local/mongodb
root@test-0003:/usr/local/mongodb# mkdir -p data/shard13
root@test-0003:/usr/local/mongodb# mkdir -p data/shard23
root@test-0003:/usr/local/mongodb# mkdir -p data/config
root@test-0003:/usr/local/mongodb# touch data/shard13.log
root@test-0003:/usr/local/mongodb# touch data/shard23.log
root@test-0003:/usr/local/mongodb#
```

分别完成shard1和shard2的replica set配置：

```
root@test-0001:/usr/local/mongodb# mongod --shardsvr --replSet shard1 --port 27017 --dbpath /usr/loc
al/mongodb/data/shard11 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard11.log --logappend -
-bind_ip=0.0.0.0 --fork
about to fork child process, waiting until server is ready for connections.
forked process: 8502
child process started successfully, parent exiting
```

```
root@test-0001:/usr/local/mongodb# mongod --shardsvr --replSet shard2 --port 27018 --dbpath /usr/local/mongodb/data/shard21 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard21.
log --logappend --bind_ip=0.0.0.0 --fork
about to fork child process, waiting until server is ready for connections.
forked process: 8612
child process started successfully, parent exiting
```

初始化shard1和shard2的replica set：

shard1，在某一台服务器上执行：

mongo 60.204.242.167:27017

config = { *id: 'shard1', members: [* {id: 0, host: '60.204.242.167:27017'}, {*id: 1, host: '60.204.228.189:27017'},* {id: 2, host: '60.204.244.116:27017'}] }

rs.initiate(config);

shard2，在某一台服务器上执行：

mongo 60.204.242.167:27018

config = { *id: 'shard2', members: [* {id: 0, host: '60.204.242.167:27018'}, {*id: 1, host: '60.204.228.189:27018'},* {id: 2, host: '60.204.244.116:27018'}] }

rs.initiate(config);

```
root@test-0001:/usr/local/mongodb# mongo 60.204.242.167:27017
MongoDB shell version v4.4.17
connecting to: mongodb://60.204.242.167:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("2498494b-f982-4229-98f7-fcfaed2591d6") }
MongoDB server version: 4.4.17
---
The server generated these startup warnings when booting:
        2023-10-30T20:31:52.990+08:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2023-10-30T20:31:53.632+08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2023-10-30T20:31:53.632+08:00: You are running this process as the root user, which is not recommended
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> config = { _id: 'shard1', members: [ {_id: 0, host: '60.204.242.167:27017'}, {_id: 1, host: '60.204.228.189:27017'}, {_id: 2, host: '60.204.244.116:27017'}] }
{
        "_id" : "shard1",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "60.204.242.167:27017"
                },
                {
                        "_id" : 1,
                        "host" : "60.204.228.189:27017"
                },
                {
                        "_id" : 2,
                        "host" : "60.204.244.116:27017"
                }
        ]
}
> rs.initiate(config);
{ "ok" : 1 }
shard1:SECONDARY>
```

```
> config = { _id: 'shard2', members: [ {_id: 0, host: '60.204.242.167:27018'}, {_id: 1, host: '60.204.228.189:27018'}, {_id: 2, host: '60.204.244.116:27018'}] }
{
        "_id" : "shard2",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "60.204.242.167:27018"
                },
                {
                        "_id" : 1,
                        "host" : "60.204.228.189:27018"
                },
                {
                        "_id" : 2,
                        "host" : "60.204.244.116:27018"
                }
        ]
}
> rs.initiate(config);
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698669536, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1698669536, 1)
}
shard2:SECONDARY>
```

配置config server：

在三台服务器上分别执行：

mongod --configsvr --replSet config --dbpath /usr/local/mongodb/data/config --port 20000 --logpath /usr/local/mongodb/data/config.log --logappend --bind_ip=0.0.0.0 --fork

在某一台服务器上执行：

mongo 60.204.242.167:20000

config = { *id: 'config', members: [* {id: 0, host: '60.204.242.167:20000'}, {*id: 1, host: '60.204.228.189:20000'},* {id: 2, host: '60.204.244.116:20000'}] }

rs.initiate(config);

```
> config = { _id: 'config', members: [ {_id: 0, host: '60.204.242.167:20000'}, {_id: 1, host: '60.204.228.189:20000'}, {_id: 2, host: '60.204.244.116:20000'}] }
{
        "_id" : "config",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "60.204.242.167:20000"
                },
                {
                        "_id" : 1,
                        "host" : "60.204.228.189:20000"
                },
                {
                        "_id" : 2,
                        "host" : "60.204.244.116:20000"
                }
        ]
}
> rs.initiate(config);
{
        "ok" : 1,
        "$gleStats" : {
                "lastOpTime" : Timestamp(1698669771, 1),
                "electionId" : ObjectId("000000000000000000000000")
        },
        "lastCommittedOpTime" : Timestamp(0, 0),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698669771, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1698669771, 1)
}
config:SECONDARY>
```

配置mongos：

在三台服务器上分别执行：

mongos --configdb config/60.204.242.167:20000,60.204.228.189:20000,60.204.244.116:20000 --port 30000 --logpath /usr/local/mongodb/data/mongos.log --logappend --bind_ip=0.0.0.0 --fork

使用mongos：

mongo 60.204.242.167:30000

切换到admin，添加分片：

use admin;

db.runCommand({addshard:"shard1/60.204.242.167:27017,60.204.228.189:27017,60.204.244.116:27017",name:"s1", maxsize:20480});

db.runCommand({addshard:"shard2/60.204.242.167:27018,60.204.228.189:27018,60.204.244.116:27018",name:"s2", maxsize:20480});

```
mongos> use admin;
switched to db admin
mongos> db.runCommand({addshard:"shard1/60.204.228.189:27017,60.204.242.167:27017, 60.204.244.116:27017",name:"s1", maxsize:20480});
{
        "ok" : 0,
        "errmsg" : "in seed list shard1/60.204.228.189:27017,60.204.242.167:27017, 60.204.244.116:27017, host  60.204.244.116:27017 does not belong to replica set shard1; found { topo
logyVersion: { processId: ObjectId('653f7c3cd1e8e41120d20c6e'), counter: 6 }, hosts: [ \"60.204.228.189:27017\", \"60.204.242.167:27017\", \"60.204.244.116:27017\" ], setName: \"shard
1\", setVersion: 1, ismaster: true, secondary: false, primary: \"60.204.228.189:27017\", me: \"60.204.228.189:27017\", electionId: ObjectId('7fffffff0000000000000001'), lastWrite: { o
pTime: { ts: Timestamp(1698663825, 1), t: 1 }, lastWriteDate: new Date(1698663825000), majorityOpTime: { ts: Timestamp(1698663825, 1), t: 1 }, majorityWriteDate: new Date(169866382500
0) }, maxBsonObjectSize: 16777216, maxMessageSizeBytes: 48000000, maxWriteBatchSize: 100000, localTime: new Date(1698663828408), logicalSessionTimeoutMinutes: 30, connectionId: 24, mi
nWireVersion: 0, maxWireVersion: 9, readOnly: false, compression: [ \"snappy\", \"zstd\", \"zlib\" ], ok: 1.0, $clusterTime: { clusterTime: Timestamp(1698663825, 1), signature: { hash
: BinData(0, 0000000000000000000000000000000000000000), keyId: 0 } }, operationTime: Timestamp(1698663825, 1) }",
        "code" : 96,
        "codeName" : "OperationFailed",
        "operationTime" : Timestamp(1698663827, 2),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698663827, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> db.runCommand({addshard:"shard2/60.204.228.189:27018,60.204.242.167:27018, 60.204.244.116:27018",name:"s2", maxsize:20480});
{
        "ok" : 0,
        "errmsg" : "in seed list shard2/60.204.228.189:27018,60.204.242.167:27018, 60.204.244.116:27018, host  60.204.244.116:27018 does not belong to replica set shard2; found { topo
logyVersion: { processId: ObjectId('653f88d203af49fdeea2c389'), counter: 6 }, hosts: [ \"60.204.228.189:27018\", \"60.204.242.167:27018\", \"60.204.244.116:27018\" ], setName: \"shard
2\", setVersion: 1, ismaster: true, secondary: false, primary: \"60.204.228.189:27018\", me: \"60.204.228.189:27018\", electionId: ObjectId('7fffffff0000000000000001'), lastWrite: { o
pTime: { ts: Timestamp(1698663830, 1), t: 1 }, lastWriteDate: new Date(1698663830000), majorityOpTime: { ts: Timestamp(1698663830, 1), t: 1 }, majorityWriteDate: new Date(169866383000
0) }, maxBsonObjectSize: 16777216, maxMessageSizeBytes: 48000000, maxWriteBatchSize: 100000, localTime: new Date(1698663838847), logicalSessionTimeoutMinutes: 30, connectionId: 23, mi
nWireVersion: 0, maxWireVersion: 9, readOnly: false, compression: [ \"snappy\", \"zstd\", \"zlib\" ], ok: 1.0, $clusterTime: { clusterTime: Timestamp(1698663830, 1), signature: { hash
: BinData(0, 0000000000000000000000000000000000000000), keyId: 0 } }, operationTime: Timestamp(1698663830, 1) }",
        "code" : 96,
        "codeName" : "OperationFailed",
        "operationTime" : Timestamp(1698663838, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698663839, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

（上图为报错，因为addshard字段内严格不允许出现空格）

```
root@test-0001:/usr/local/mongodb# mongo 60.204.242.167:30000
MongoDB shell version v4.4.17
connecting to: mongodb://60.204.242.167:30000/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d7addd2e-c143-4e61-87f5-45ff525820b8") }
MongoDB server version: 4.4.17
---
The server generated these startup warnings when booting:
        2023-10-30T20:44:18.813+08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2023-10-30T20:44:18.813+08:00: You are running this process as the root user, which is not recommended
---
mongos> use admin;
switched to db admin
mongos> db.runCommand({addshard:"shard1/60.204.242.167:27017,60.204.228.189:27017,60.204.244.116:27017",name:"s1", maxsize:20480});
{
        "shardAdded" : "s1",
        "ok" : 1,
        "operationTime" : Timestamp(1698669951, 5),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698669951, 5),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> db.runCommand({addshard:"shard2/60.204.242.167:27018,60.204.228.189:27018,60.204.244.116:27018",name:"s2", maxsize:20480});
{
        "shardAdded" : "s2",
        "ok" : 1,
        "operationTime" : Timestamp(1698669965, 3),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698669965, 3),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos>
```

激活数据库分片：

创建一个名为testdb的数据库：

use testdb

```
mongos> use testdb
switched to db testdb
```

激活分片：

sh.enableSharding("testdb")

```
mongos> sh.enableSharding("testdb")
{
        "ok" : 1,
        "operationTime" : Timestamp(1698670042, 8),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698670042, 8),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos>
```

使用sh.status()查看数据库当前情况：

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("653fa4d673838d5e44d4bc05")
  }
  shards:
        {  "_id" : "s1",  "host" : "shard1/60.204.228.189:27017,60.204.242.167:27017,60.204.244.116:27017",  "state" : 1 }
        {  "_id" : "s2",  "host" : "shard2/60.204.228.189:27018,60.204.242.167:27018,60.204.244.116:27018",  "state" : 1 }
  active mongoses:
        "4.4.17" : 3
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                28 : Success
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
                config.system.sessions
                        shard key: { "_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                s1      996
                                s2      28
                        too many chunks to print, use verbose if you want to force print
        {  "_id" : "testdb",  "primary" : "s2",  "partitioned" : true,  "version" : {  "uuid" : UUID("1b3b503d-a2b2-4e9c-b435-be52f5d5c4dc"),  "lastMod" : 1 } }
mongos>
```

创建一个新的集合testc：

db.createCollection("testc")

```
mongos> db.createCollection("testc")
{
        "ok" : 1,
        "operationTime" : Timestamp(1698673262, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1698673262, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

插入一些数据

db.testc.insertMany([
 {"*id": 1, "name": "LJP", "age": 27},*
 {"id": 2, "name": "DJE", "age": 42},
 {"*id": 3, "name": "OFW", "age": 61},*
 {"id": 4, "name": "SLX", "age": 15}
]);

```
mongos> db.testc.insertMany([
...     {"_id": 1, "name": "LJP", "age": 27},
...     {"_id": 2, "name": "DJE", "age": 42},
...     {"_id": 3, "name": "OFW", "age": 61},
...     {"_id": 4, "name": "SLX", "age": 15}
... ]);
{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4 ] }
mongos>
```

通过db.testc.find()可以查看当前集合中的数据：

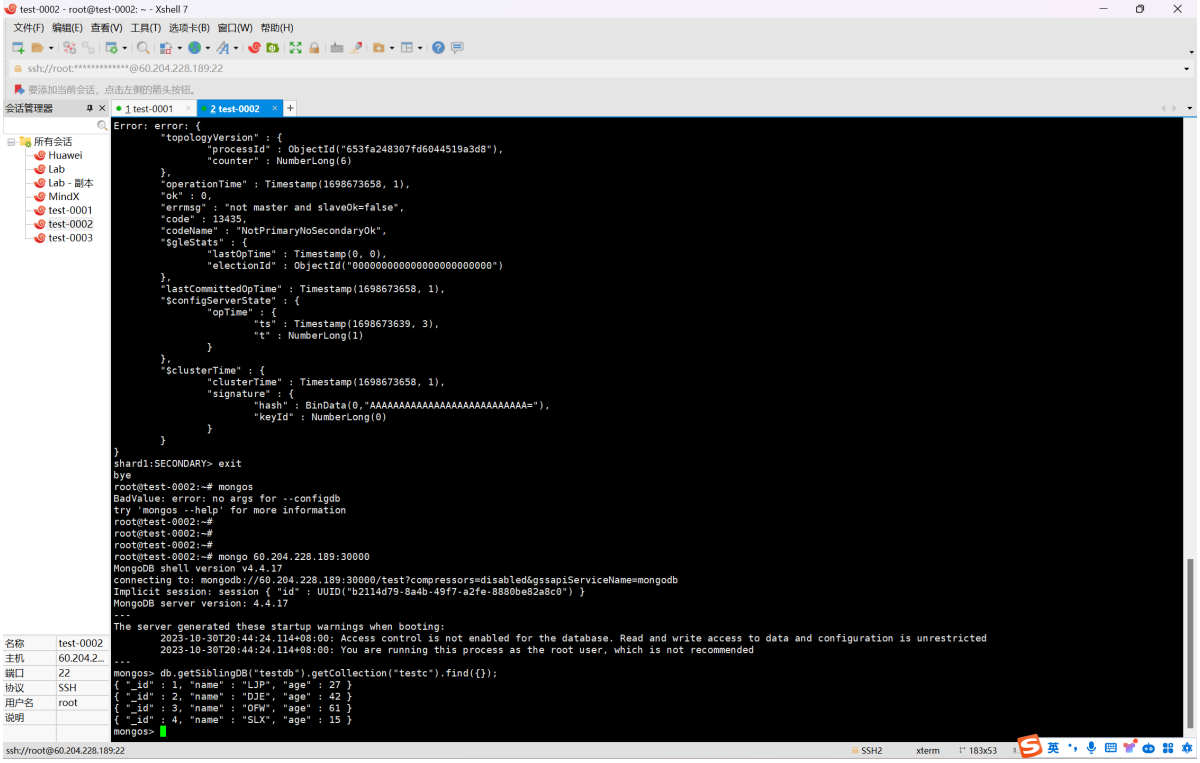
```
mongos> db.testc.find()
{ "_id" : 1, "name" : "LJP", "age" : 27 }
{ "_id" : 2, "name" : "DJE", "age" : 42 }
{ "_id" : 3, "name" : "OFW", "age" : 61 }
{ "_id" : 4, "name" : "SLX", "age" : 15 }
```

此时打开test-0002服务器进入mongos：

mongo 60.204.228.189:30000

查询数据：

db.getSiblingDB("testdb").getCollection("testc").find({});



删除一条数据：

db.getSiblingDB("testdb").getCollection("testc").deleteOne({ _id: 3 })

```
mongos> db.getSiblingDB("testdb").getCollection("testc").deleteOne({ _id:3 })
{ "acknowledged" : true, "deletedCount" : 1 }
```

再回到test-0001服务器，通过db.testc.find()查看当前集合中的数据：

```
mongos> db.testc.find()
{ "_id" : 1, "name" : "LJP", "age" : 27 }
{ "_id" : 2, "name" : "DJE", "age" : 42 }
{ "_id" : 4, "name" : "SLX", "age" : 15 }
```

MySQL和mongoDB的MVCC对比分析：

MySQL和MongoDB都实现了MVCC（多版本并发控制）机制，用以解决读写冲突的并发控制。在MVCC机制中，为事务分配单向增长的时间戳，并为每个修改保存一个版本，版本与事务时间戳关联。读操作只会读取该事务开始前的数据库快照，从而避免阻塞其它读操作。

虽然二者都采用了MVCC，但在具体实现上存在显著差异。MySQL的MVCC是通过保存数据的历史版本来实现的，每个数据项都有一个时间戳timestamp，可以追踪到创建和修改的时间点。这种实现方式使得MySQL能够提供严格的可重复读，保证读取操作不会看到未提交（commit）的修改，对已修改但未提交的数据进行更新也是非法的。

MongoDB的MVCC实现更复杂。在MongoDB中，每个文档都有一个包含多个版本的时间戳数组，也可以看作一个topologyVersion（拓扑版本号）。当文档被修改时，旧版本并不会被删除，而是存储在数组中，而在绝大部分情况下读写操作都使用的是当前最新的版本（当然，这种方式也允许MongoDB在读取数据时查看过时的数据版本），从而实现非阻塞读写操作。